

	C	D	E	F
C, D, E, F			(	<=
1			^	<>
2			*	=>
3			+	=<
4	SQRT			<
5			-	=
6	INT			>
7			/	NOT
8		FREE		
9	INP	INP		
A	SGN	PEEK		
B	SIN	ABS		
C	LEN	COS	AND	
D	CALL	LOG	OR	
E	RND	EXP		
F		TIME	>=	
				end of lower table

BASIC's
machine code

Machine
Code

BASIC

2000	C3 41 20	JMP 2041H	COLD START
2003	C3 43 20	JMP 2043H	WARM START
2006	1A	DB CTL Z	FRNT PANEL ENTRY CODE
2007	1B	DB ESC	ESCAPE CODE (CTL-Y)
2008	50	DB 'P'	TAPE MODE POLYHASE OR BYTE
2009	3E	DP '>'	PROMPT CHARACTER
200A	21 20 22 0C	ASC '!@'F'	Start up Message written in BASIC
200E	50 6F 6C 79	ASC 'Poly'	
2012	20 38 38 20	ASC '88'	
2016	42 41 53 49	ASC 'BASI'	
201A	43 20 76 65	ASC 'Cve'	
201E	72 73 69 6F	ASC 'rsio'	
2022	6E 20 41 30	ASC 'nA'	
2026	30 2E 22 2C	ASC '0."'	
202A	46 52 45 45	ASC 'FREE'	
202E	28 30 29 2C	ASC '(0)'	
2032	22 20 62 79	ASC '"by'	
2036	74 65 73 20	ASC 'tes'	
203A	66 72 65 65	ASC 'free'	
203E	2E 22 00	ASC '."R'	
2041	AF	XRA A	2041 sets Z only
2042	2E 37	MVI L, 37H	2043 sets C only
2043		STC	
2044	31 00 10	LXI SP, 1000H	SETUP STACK PTR
2047	CD A0 20	CALL 20A0H	CALL CANCEL RTN

Q544-0045

204A	21 69 20	LXI H, 2069H	KYBD INT RTN
204D	22 18 0C	SHLD 0C18H	SRA5 - KYBD INT.
2050	21 64 00	LXI H, 064H	LXI H, 10RET
2053	22 9E 48	SHLD 489EH	
2056	21 BA 20	LXI H, 20BAH	BASIC's WH0 r/n
2059	22 21 0C	SHLD WH0+1	
205C	21 ⁵⁰ 00 ⁴⁹ 4B	LXI H, TOP OF BASIC	
205F	11 FF 4F	LXI D, 4FFFFH	
2062	CD F6 20	CALL 20F6H	find top of memory?
2065	C3 18 21	JMP 2118H	
2068	21 ?	Copy right byte?	
2069	DB F8	BASICS KEYBOARD INTERRUPT SERVICE RTN	
		see next page	

2045H
8267H
69/20
6F/45
" 69

BASIC'S KEYBOARD INTERRUPT HANDLER

2069	DB F8	IN 0F8H	
206B	FE 18	CPI CANCEL	
206D	CC A0 20	CZ 20A0H	DO CANCEL RTN
2070	47	MOV B,A	
2071	3A 07 20	LDA 2007H	ESCAPE CODE
2074	B8	CMP B	
2075	CA A0 20	CZ 20A0H	DO ESCAPE RTN
2078	3A 06 20	LDA 2006H	FRONT PANEL ENTRY CODE (CTL-2)
207B	B8	CMP B	
207C	CA 9A 20	CZ 209AH	EXIT ROUTINE
207F	2A A2 48	LHL D 48A2H	PUT CHAR PTR INTO HL
2082	70	MOV M,B	PUT CHAR AT PTR
2083	2B	DCX H	
2084	3E 7F	MVI A, 07FH	
2086	BD	CMP L	
2087	C2 80 20	JNZ 2080H	
208A	21 BF 0C	LXI H, CBFH	
208D	3A A0 48	LDA 48A0H	
2090	BD	CMP L	
2091	CA C4 00	JZ 064 (IORET)	
2094	22 A2 48	SHLD 48A2H	
2097	C3 64 00	JMP 064H (IORET)	

CTL-2 RTN

209A	CD A0 20	CALL 20A0H	DO CANCEL ROUTINE
209D	C3 17 01	JMP 117H	S.S. INT.

CANCEL RTN

20A0	F3	OI	← CANCEL RTN
------	----	----	--------------

20A1	E5	PUSH H
20A2	21 BF 0C	LXI H, 0CBFH
20A5	22 A2 48	SHLD 48A2H
20A8	22 A0 48	SHLD 48A0H
20AB	E1	POP H
20AC	C9	RET

ESCAPE RTN or CTL-Y

20AD	CD A0 20	CALL 20A0H	DO CANCEL RTN
20B0	32 9D 48	STA 489DH	
20B3	2A 9E 48	LHLD 489EH	usually holding 10RET
20B6	E9	PCHL	

BASICs WH0

20B7	E1	POP H	
20B8	FB	EI	
20B9	76	HLT	
BASICs WH0 → 20BA	F3	DI	NORMAL ENTRY POINT
20BB	E5	PUSH H	
20BC	2A A0 48	LHLD 48A0H	
20BF	3A A2 48	LDA 48A2H	
20C2	BD	CMP L	
20C3	CA B7 20	JZ 20B7H	JMP IF EQUAL
20C6	7E	MOV A, m	
20C7	F5	PUSH PSW	
20C8	2B	DCX H	
20C9	3E 7F	MVI A, 07FH	(OCL)
20CB	DB	CMP L	
20CC	C2 D2 20	JNZ 20D2H	JMP IF NOT EQUAL

20CF 21 BF 0C
 20D2 22 A0 48
 20D5 F1
 20D6 FB
 20D7 2A B2 48
 20DA B3 HL
 20DB C9

LXI H, 0CBFH
 SHLD 48A0H
 POP PSW

EI

LHL 48B2H → contains flip or no flip routine
 3A9

XTHL

RET

so, put 3A9 at TOP of STACK
 pop 3A9 off stack goto 3A9
 at 3A9 is a return also

{ 3A9 if 'not flipped'
 { 20DC if 'flipped'

3A9 = RET

20DC on next page

X

flip processing

200C	FE 40	CPI '@'	
200E	D8	RC	uppercase RETURN IF Accum < @
200F	FE 5B	CPI '['	CARRY SET @A...XYZ [] ^ _ 'a b c (etc)
20E1	FA EA 20	JM 20EAH	JMP IF A is {A-Z}
20E4	FE 61	CPI 'a'	
20E6	D8	RC	lower case RETURN IF CARRY
20E7	FE 7B	CPI '{'	xyz { } (etc)
20E9	D0	RNC	RET if '{'
20EA	EE 20	XRI 20H	XOR A with 20H
20EC	C9	RET	complement bits
20ED	78	MOV A, B	
20EE	FE 18	CPI CAN	
20F0	CC A0 20	CZ 20A0H	CALL IF zero CANCEL RTN
20F3	C3 24 0C	JMP WH1	
20F6	EB	XCHG	(TOP OF MEMORY FINDER?)
20F7	F5	PUSH PSW	
20F8	7E	MOV A, m	- HL = start addr, DE = top of BASIC
20F9	2F	CMA	
20FA	77	MOV m, A	
20FB	BE	CMP m	
20FC	C2 00 21	JNZ 2100H	
20FF	2F	CMA	
2100	77	MOV m, A	
2101	BE	CMP m	
2102	C2 00 21	JNZ 2100H	

2105	23	INX H	
2106	3A 1F 0C	LDA 0C1FH	SCRHM
2109	BC	CMP H	check if TOP of memory = screen
210A	C2 F8 20	JNZ 20F8H	

20PC
2102 → 210D 32 A8 49 STA 49A8H FOUND TOP

2110	7E	MOV A, M
2111	32 A9 49	STA 49A9H
2114	2B	DCX H
2115	EB	XCHG
2116	F1	POP PSW
2117	C9	RET

START
AFTER
2000
or
2003 → 2118 23 2 INX H

2119	22 AC 49	SHLD 49ACH	start of text
211C	EB	XCHG	

211D	22 AE 49	SHLD 49AEH	Top of mem
------	----------	------------	------------

2120	21 A9 03	LXI H, 3A9	or 20DC for auto-ftp
------	----------	------------	----------------------

2123	22 B2 48	SHLD 48B2
------	----------	-----------

2126	21 00 10	LXI H, 1000H
------	----------	--------------

2129	F9	SPHL	set SP for to 1000H
------	----	------	---------------------

212A	22 B0 48	SHLD 48B0
------	----------	-----------

212D	D2 48 21	JNC 2148H
------	----------	-----------

2130	2A AC 49	LHLD 49ACH
------	----------	------------

213C → 2133 7E MOV A, M

2134	FE 01	CPI 1
------	-------	-------

2136	CA 3F 21	JZ 213FH
------	----------	----------

2139	CD 51 45	CALL 4551H
------	----------	------------

ADD L
MOV L, A
RNC

INR H
RET

213C C3 33 21

JMP 2133H

2136 → 213F 22 AA 48

SHLD 48AAH

store top of BASIC text

2142 CD 20 25

CALL 2520H

CLEAR routine

2145 C3 4B 21

JMP 214BH

212D → 2148 CD 18 25

CALL 2518H

= SCR routine & CLEAR

→ 214B AF

XRA A

make A = 0

214C 32 A7 49

STA 49A7H

if 49A7 = 00, then execute startup routine

214F CD 33 3A

CALL 3A33H

LXI H, 2300H

SHLD 496BH

RET

2152 CD 19 3A

CALL 3A19H

LHLD 496BH

SHLD 496FH

LHLD 496D

LXI H, 0001

SHLD 495D

RET

→ SHLD 4971
RET

2155 CD AB 35

CALL 35AB

FROM
214D →
2240
2241
2282

2158 2A B0 48

LHLD 48B0H

215B F9

SPHL

make (HL) new stack ptr

215C CD D1 3D

CALL 3DD1H

returns if 48AB = 0, else put CR into B,
print it and store a 0 at 48AG

215F 3E 01

MVI A, 1

2161 32 AC 48

STA 48ACH

2164 21 58 21

LXI H, 2158H

2167 22 9E 48

SHLD 489EH

216A FB

EI

216B 21 00 00

LXI H, 0

216E 22 A1 49

SHLD 49A1H

2171 22 A3 49

SHLD 49A3H

2174 AF

XRA A

make A = 0

FROM
21A7 →

2175 32 9D 48

STA 489DH

2178 3A 09 20

LDA PROMPT

217B B7

ORA A

set flags

217C CA 8D 21

JZ 218DH

HEEIS QMT

513C C3 33 51

0 1 2 3 4 5 6 7 8 9 A B C D E F

513E 55 AA 48

HEEIS QMT

5145 CD 50 52

8 4 2 514B 51

5142 C3 48 51

5148 CD 18 52

E = 1110

A = 1010

514B AF

HEEIS QMT

514C 35 A3 44

8 4 21

514E CD 33 3A

HEEIS QMT

5125 CD 14 3A

HEEIS QMT

5122 CD A6 33

HEEIS QMT

5128 5A B0 48

HEEIS QMT

512B CD 10 30

HEEIS QMT

512C CD 01 30

HEEIS QMT

512F 3E 07

HEEIS QMT

5161 35 AC 48

HEEIS QMT

5164 51 28 51

HEEIS QMT

5165 55 AC 48

HEEIS QMT

516A 6B

HEEIS QMT

516B 51 00 00

HEEIS QMT

516E 55 A3 44

HEEIS QMT

5171 55 A3 44

HEEIS QMT

5174 AF

HEEIS QMT

5175 35 40 48

HEEIS QMT

5178 3A 00 50

HEEIS QMT

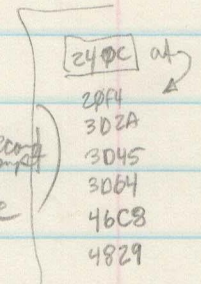
517B 65

HEEIS QMT

517C CA 80 51

3DA7 has startup prompt
set up

	217F	47	MOV B,A	
	2180	CD B9 3D	CALL 3DB9H	→ does a cancel then outputs B via WH1
	2183	3A AD 48	LDA 48ADH	
	2186	B7	ORA A	set flags
	2187	CA 8D 21	JZ 218DH	
	218A	CD B9 3D	CALL 3DB9H	→ output second prompt
217C →	218D	CD E9 3C	CALL 3CE9H	→ Get a command or CR
	2190	CD D6 3D	CALL 3DD6H	→ print a CR
	2193	3A EA 48	LDA 48EAH	START OF UNCOMPRESSED INPUT BUFFER
	2196	FE 00	CPI CR	
	2198	CA 78 21	JZ 2178H	no input if a CR at 48EA
	219B	CD EC 23	CALL 23ECH	→ input compressor
DEB or 55 here	219E	DA AA 21	JC 21AAH	→ execute a command or statement
↑	21A1	CD 9B 44	CALL 449BH	→ how to do with end of text
	21A4	CD 20 25	CALL 2520H	→ CLE routine
	21A7	C3 78 21	JMP 2178H	
FROM 219E →	21AA	CD B0 21	CALL 21B0H	
	21AD	C3 58 21	JMP 2158H	
FROM 21AA →	21B0	21 C2 48	LXI H, 48C2H	Command Input Execution
	21B3	22 B4 48	SHLD 48B4H	
	21B6	CD 97 39	CALL 3997H	→ advance to first char
	21B9	EG E0	ANI E0H	(or ' ' + 128) = zero all bits except for MSB
	21BB	FE A0	CPI A0H	(or 'B' + 128) = RUN
	21BD	11 3B 2E	LXI D, 2E3BH	→ where RUN DW is
	21C0	CA E1 26	JZ 26E1H	→ if a command
	21C3	CD C9 26	CALL 26C9H	→ if a statement



	21C6	CD 97 39	CALL 3997H	
	21C9	FE 0D	CPI CR	
	21CB	C8	RZ	return if CR
	21CC	21 A9 48	LXI H, 48A9H	
	21CF	BE	CMP M	
	21D0	C8	RZ	return if contents of 48A9 = Accumulator
4665 →	21D1	01 33 23	LXI B, 2333H	Syntax"
	21D4	C3 20 22	JMP 2220H	
	21D7	01 D0 22	LXI B, 22D0H	Subscript"
	21DA	C3 20 22	JMP 2220H	
2575 2564 2574 →	21DD	01 DA 22	LXI B, 22DAH	Bad Argument"
	21E0	C3 20 22	JMP 2220H	
	21E3	01 E7 22	LXI B, 22E7H	Dimension"
	21E6	C3 20 22	JMP 2220H	
	21E9	01 F1 22	LXI B, 22F1H	Function Def"
	21EC	C3 20 22	JMP 2220H	
2569 3BDD 3BEB 3BEB →	21EF	01 FE 22	LXI B, 22FEH	Out of Bounds"
	21F2	C3 20 22	JMP 2220H	
	21F5	01 0C 23	LXI B, 230CH	Type"
	21F8	C3 20 22	JMP 2220H	
	21FB	01 11 23	LXI B, 2311H	Format"
	21FE	C3 20 22	JMP 2220H	
	2201	01 18 23	LXI B, 2318H	Line number"
	2204	C3 20 22	JMP 2220H	
	2207	01 02 23	LXI B, 2302	For-Next
	220A	C3 20 22	JMP 2220H	

224B

2200 01 B0 23 LXI B, 23B0 RETURN without COSUB

2210 C3 20 22 JMP 2220H

2213 2A 4A 48 LHLD 48AA

2216 2B DCX H

2217 22 B4 48 SHLD 48B4

221A C3 20 22 JMP 2220H

2210 01 24 23 LXI B, 2324H Divide by 0"

2220 C0 A0 20 CALL 20A0H CANCEL RTN

2223 FB EI -just in case

2224 76 HLT BC=error msg, HL=error inline, DE=char count of

2225 3A A8 48 LDA 48A8H

2228 AF XRA A

2229 CA 3B 22 JZ 223BH make A=0
if executed XRAA, always jump

222C 2A 69 49 LHLD 4969H

222F 22 B8 48 SHLD 48B8H

2232 2A 67 49 LHLD 4967H

2235 11 00 00 LXI B, 0

2238 C0 F8 36 CALL 36F8H

2229 223B C5 PUSH B error msg ptr.

223C C0 D1 30 CALL 30D1H

223F 3A 90 48 LDA 4890H

2242 B7 ORA A set flags

2243 C2 81 22 JNZ 2281H

2246 32 AB 49 STA 49ABH

2249 3A AC 48 LDA 48ACH

224C B7 ORA A set flags

224D	C2 81 22	JNZ 2281H	
2250	CD BB 22	CALL 22BBH	
2253	60	MOV H, B	} PUT B, C INTO H, L
2254	69	MOV L, C	
2255	11 C2 48	LXI D, 48C2H	
2258	CD AF 24	CALL 24AFH	expand line for error display
225B	CD D6 3D	CALL 3DD6H	print a CR
225E	21 C2 48	LXI H, 48C2H	
2261	CD C4 39	* CALL 39C4H	print error line
2264	CD D6 3D	CALL 3DD6H	print a CR
2267	3A AB 49	LDA 49ABH	
226A	E6 3F	ANI '?'	
226C	CA 81 22	JZ 2281H	
226F	57	MOV D, A	
2270	06 20	MVI B, 'B'	
2272	CD ED 2D	CALL 2DEDH	
2275	15	DCR D	
2276	C2 70 22	JNZ 2270H	- output char.
2279	06 9C	MVI B, 9CH "↑"	
227B	CD ED 2D	CALL 2DEDH	
227E	CD D6 3D	CALL 3DD6H	print a CR
2281	E1	POP H	
2282	CD C9 39	CALL 39C9H	
2285	21 14 2E	LXI H, 2E14H	"Error"
2288	FB	EI	
2289	CD C9 39	CALL 39C9H	

Handwritten signature/initials on the right side of the page.

228C	3A AC 48	LDA 48ACH	
228F	B7	ORA A	set flags
2290	C2 58 21	JNZ 2158H	
2293	3A 9D 48	LDA 489DH	
2296	B7	ORA A	set flags again
2297	CA 58 21	JZ 2158H	
229A	21 1B 2E	LXI H, 2E1BH	where of ERROR IN LINE (line#)
229D	CD C9 39	CALL 39C9H	print IN LINE is to be printed
22A0	CD BB 22	CALL 22BBH	find error line #
22A3	03	INX B	
22A4	0A	LDAX B	
22A5	6F	MOV L, A	
22A6	03	ANA M INX B	
22A7	0A	LDAX B	
22A8	67	MOV H, A	put line # into HL
22A9	11 C2 48	LXI D, 48C2H	
22AC	CD AC 3C	CALL 3CACH	Make LN ASCII decimal for ERROR IN (line#)
22AF	3E 0D	MVI A, CR	
22B1	12	STAX D	put a CR at end of error message
22B2	21 C2 48	LXI H, 48C2H	
22B5	CD C4 39	CALL 39C4H	MVI C, CR JMP 39CB print error message
22B8	C3 58 21	JMP 2158H	
22BB	2A AC 49	LHLD 49ACH	start of test DW. [find error line]
22BE	44	MOV B, H	} MOVE HL INTO BC
22BF	4D	MOV C, L	
22C0	5E	MOV E, M	

22BB	24 AC 49	CHLD 49ACH	start of test DW
22BE	44	MOV B, H	
22BF	4D	MOV C, L	
22C0	5E	MOV E, M	
22C1	16 00	MVI D, 0	
22C3	19	DAD D	HL = DE + HL
22C4	EB	XCHG	put result into DE
22C5	21 B4 48	LXI H, 48B4H	current char DW
22C8	CD 0B 39	CALL 390B ⁴	compare to see if equal?
22CB	EB <u>DBP</u>	XCHG <u>390B</u>	cmp DE w/ contents of 48B4
22CC	00	RNC	cmp
22CD	C3 BE 22	JMP 22BEH	

22D0	53 75 62 73 63 72 69 70 74 22	"Subscript"
22DA	42 61 64 20 61 72 67 75 6D 65 6E 74 22	"Bad argument"
22E7	44 69 6D 65 6E 73 69 6F 6E 22	"Dimension"
22F1	46 75 6E 63 74 69 6F 6E 20 44 65 66 22	"Function Def"
22FE	4F 75 74 20 6F 66 20 62 6F 75 6E 64 73 22	"Out of Bounds"
230C	54 79 70 65 22	"Type"
2311	46 6F 72 6D 61 74 22	"Format"
2318	4C 69 6E 65 20 6E 75 6D 62 65 72 22	"Line number"
2324	44 69 76 69 64 65 20 62 79 20 74 65 72 65 22	"Divide by zero"
2333	53 79 6E 74 61 78 22	"Syntax"
233A	43 61 6E 27 74 20 63 6F 6E 74 69 6E 22	"Can't continue"
2349	44 6F 75 62 6C 65 20 64 65 66 22	"Double def"
2353	43 6F 6E 74 72 6F 6C 20 73 74 61 63 6B 22	"Control stack"
2362	52 65 61 64 22	"Read" (should be READ)
2367	43 6F 6D 70 6C 65 78 69 74 79 22	"Complexity"
2372	49 6E 70 75 74 22	"Input"
2378	4D 65 6D 6F 72 79 20 66 75 6C 6C 22	"Memory full"
2384	41 72 67 20 6D 69 73 6D 61 74 63 68 22	"Arg mismatch"

2391 49 6C 6C 65 67 61 6C 20 64 69 72 65 63 74 22

Illegal / Direct"

23A0 4C 65 6E 67 74 68 22

Length"

23A7 4F 76 65 72 66 6C 6F 77 22

Overflow"

23B0 52 45 54 55 52 4E 20 77 69 74 68 6F 75 74 20 47 4F 42 22 ^{53 55} RETURN without GOSUB"

23C5 4D 69 73 73 69 6E 67 20 4E 45 58 54 22

Missing NEXT"

23D2 46 4F 52 2D 4E 45 58 54 22

FOR-NEXT"

23DB 43 68 65 63 6B 73 75 6D 22

Checksum"

23E4 56 65 72 69 66 79 22

Verify"

Input compressor

23EB 06 ~~06 06 06 06~~ why?

219B → 23EC 21 EA 48 LXI H, 48EAH - first byte of uncompressed input

23EF 22 B4 48 SHLD 48B4H

calls 3997 / stores new HL at 497E / calls 4455
which checks for minuses
gets line #

23F2 CD 60 3C CALL 3C60H

put line # at start of line + 1

23F5 22 C0 48 SHLD 48C0H

23F8 F5 PUSH PSW

save A and flags

23F9 2A B4 48 LHLD 48B4H - start of uncompressed input ptr

23FC 0E 04 MVI C, 4

23FE 11 C2 48 LXI D, 48C2H compressed input buffer

2432 → 2468 → 2480 → 2401 CD A7 24 CALL 24A7H moves HL(m) to DE(m) when p else stops after storing non-blank

2404 D5 PUSH DSW save compressed input buffer

2405 11 7F 2E LXI D, 2E7FH - start of token look up table

2408 ES PUSH H

2409 1A LDAX D

2E7F HAS AN 80H

240A 47 MOV B, A

240B 13 INX D

240C 1A LDAX D

LDX D, TOKEN TABLE - 2E7FH

2408 E5
2409 1A
240A 47
240B 1B
240C 1A
240D BE

PUSH H
LDAX D
MOV B, A - put token into B
INX D
LDAX D
CMP M

240E C2 15 24
2411 23
2412 C3 0B 24

JNZ 2415H
INX H
JMP 240BH

2415 B7
2416 FA 47 24

ORA A
JM 2447H

set flags

JMP if over 80H - then at end of 1 expected token

2419 43
241A 1A
241B B7

INX D
LDAX D
ORA A

241C F2 19 24

JP 2419H

JMP if < 80

241F E1

POP H

2420 EE FF

XRI 0FFH

see if at end of table

2422 C2 08 24

JNZ 2408

* BELOW HERE IS NO MATCH W/ TABLE

2425 01

POP D

2426 7E

MOV A, M

2427 FE 00

CPI Cr

2429 CA 9B 24

JZ 249BH

242C 12

STAX D

242D 13

INX D

242E 0C

INR C

242F 23

INX H

2430 FE 22

CPI ''

2432 C2 01 24

JNZ 2401H

2435 7E

MOV A, M

2436 FE 00

CPI Cr

2438 CA 9B 24

JZ 249BH

2444

243B	12	—	—	STAX D
243C	13			INX D
243D	0C			INR C
243E	23			INX H
243F	FE	22		CPI ""
2441	CA	01	24	JZ 2401H
2444	C3	35	24	JMP 2435H

(2416) → 2447	F1			POP PSW
2448	D1			POP D
2449	78			MOV A,B tokens saved at 240A - compared ok
244A	12			STAX D
244B	13			INX D
244C	0C			INR C
244D	FE	89		CPI GOSUB
244F	CA	6B	24	JZ 246BH
2452	FE	88		CPI GOTO
2454	CA	6B	24	JZ 246BH
2457	FE	96		CPI EXIT
2459	CA	6B	24	JZ 246BH
245C	FE	8E		CPI RESTORE
245E	CA	6B	24	JZ 246BH
2461	FE	9B		CPI ELSE
2463	CA	6B	24	JZ 246BH
2466	FE	9D		CPI THEN
2468	C2	01	24	JNZ 2401H
(2498) → 246B	CD	A7	24	CALL 24A7H

Tokens which might precede a line # so we might need a 97H

246E	22 B4 48	SHLD 48B4H	
2471	C5	PUSH B	
2472	D5	PUSH D	
2473	CD 60 3C	CALL 3C60H	
2476	D1	POP D	
2477	C1	POP B	
2478	3E 97	MVI A, 97H	line # header
247A	12	STAX D	
247B	44	MOV B, H	
247C	7D	MOV A, L	
247D	2A B4 48	LHLD 48B4H	
2480	DA 01 24	JC 2401H	
2483	13	INX D	
2484	0C	INR C	
2485	12	STAX D	
2486	13	INX D	
2487	0C	INR C	
2488	78	MOV A, B	
2489	12	STAX D	
248A	13	INX D	
248B	0C	INR C	
248C	CD A7 24	CALL 24A7H	
248F	FE 2C	CPI ','	
2491	C2 04 24	JNZ 2404H	
2494	12	STAX D	
2495	13	INX D	

2496	23	INX	H	
2497	0C	INR	C	
2498	C3 6B 24	JMP	246BH	
249B	3E 0D	MVI	A, CR	
249D	12	STAX	D	DE = end of line
249E	21 BF 48	LXI	H, 48BFH	very start of line ^{48C2 = start of program line-hold}
24A1	71	MOV	M, C	- length of line that was input
24A2	F1	POP	PSW	
24A3	C9	RET		
24A4	23	INX	H	- Used when compressing text -
24A5	13	INX	D	MOVES MEMORY POINTED TO BY
24A6	0C	INR	C	HL TO MEMORY POINTED TO
24A7	7E	MOV	A, M	DE AS LONG AS (HL) = 0
24A8	12	STAX	D	ELSE STORES NON BLANK CHAR
24A9	FE 20	CPI	'0'	AND RETURNS
24AB	CA A4 24	JZ	24A4H	C COUNTS # of CHARS.
24AE	C9	RET		
24AF	23	INX	H	for 1st ^{line} + expand tokens use *
24B0	E5	PUSH	H	HL = line, DE = where to put expanded line
24B1	CD E3 39	CALL	39E3H	LHL is the DW pointed to by HL
24B4	CD AC 3C	CALL	3CACH	change line# to ASCII as put at DE on
2515 → 24B7	E1	POP	H	
2516 → 24B8	23	INX	H	
24B9	23	INX	H	
24BA	3A B4 48	LDA	48B4H	
24BD	95	SUB	L	A = A - L

*

250A

INX H
LDA 48B4
SUB L

HL = where we are getting ch
DE = where we are putting ch

24B9 23
24BA 3A B4 48
24BD 95

24BE 3A B5 48 LDA 48B5H

24C1 9C SBB H

24C2 02 04 24 JNC 24D4H

24C5 3A AB 49 LDA 49ABH

24C8 B7 ORA A set flags

24C9 C2 04 24 JNZ 24D4H

24CC 7B MOV A, E

24CD 06 C3 SUI 0C3H

24CF E6 3F ANI 3FH '?'

24D1 32 AB 49 STA 49ABH

24D4 7E MOV A, M

24D5 FE 97 CPI fine# header

24D7 CA 00 25 JZ 2500H

24DA B7 ORA A set flags

24DB FA F4 24 JM 24F4H see next page expand token

24DE 12 STAX D

24DF FE 00 CPI CR

24E1 C2 F0 24 JNZ 24F0H

24E4 3A AB 49 LDA 49ABH

24E7 B7 ORA A set flags

24E8 C0 RNZ

24E9 7B MOV A, E

24EA 06 C3 SUI 0C3H

24EC 32 AB 49 STA 49ABH

24EF C9 RET

24F0 13 INX D
24F1 C3 B9 24 JMP 24B9 ~ w/p 16 page

A = taken char
HL = where we
got char from
DE = what we
expect char
at

A has ~~TOKEN~~ to
be found

CHANGES 1 BYTE
CODE TO FULL
WORD

240B → 24F4 E5	PUSH H	- save current text ptr
24F5 21 7F 2E	LXI H, 2E7FH	START OF STATEMENT TABLE
24F8 BE	CMP M	COMPARE A with (HL) CONTENTS POINTS TO (B-1) HL
24F9 23	INX H	INCREMENT TO NEXT CHAR
24FA C2 F8 24	JNZ 24F8H	JUMP IF NOT EQUAL
24FD 7E	MOV A, M	get next char ^{after} match into A
24FE B7	ORA A	set flags
24FF FA 09 25	JM 2509H	- were done → JUMP IF GREATER THAN 8DH WE'RE DONE
2502 12	STAX D	STORE 1 BYTE STATEMENT AT (DE)
2503 13	INX D	INCREMENT DE = put ptr.
2504 23	INX H	" HL = get ptr.
2505 C3 FD 24	JMP 24FDH	DO SOME MORE
2508 1E → E1	MVI E, 0E4H	? COPYRIGHT ? BYTE RESTORE H FINISHED HERE
2509 E1	POP H	
250A C3 B9 24	JMP 24B9H	EXIT
250D 23	INX H	
250E E5	PUSH H	
250F CD E3 39	CALL 39E3H	← 4HLD's (HL)
2512 CD AC 3C	CALL 3CACH	expand HL to ASCII decimal put at (DE) on
2515 C3 B7 24	JMP 24B7H	
2518 24 AC 49	LHLD 49ACH	} SCR
251B 36 01	MVI M, 1	
251D 22 AA 48	SHLD 48AAH	
2520 2A AE 49	LHLD 49AEH	TOP OF MEM DW
2523 36 00	MVI M, 0	ZERO TOP OF MEM
2525 2B	DCX H	DCR TO NEXT LOWER BYTE

21AH
2102

2526	22	5F	49	SHLD	495FH	
2529	2A	AA	48	LHLD	48AAH	- top of text ptr.
252C	23			INX	H	
252D	22	B8	48	SHLD	48B8H	
2530	21	34	00	LXI	H, 034H	
2533	CD	51	39	CALL	3951H	(*)
2536	AF			XRA	A	make A=0
2537	32	AD	48	STA	48ADH	
253A	C9			RET		

253B	21	0C	20	LXI	H, 200CH	flip routine
253E	22	B2	48	SHLD	48B2H	
2541	C9			RET		

2540 → 2542	23			INX	H	Increment HL past line length len.
2543	23			INX	H	" " " " > number
2544	23			INX	H	" " " "
2545	22	B4	48	SHLD	48B4H	
2548	C9			RET		

2549	21	0A	00	LXI	H, 0AH	
254C	22	A1	49	SHLD	49A1H	
254F	22	A3	49	SHLD	49A3H	
2552	CD	60	3C	CALL	3C60H	
2555	DA	6F	25	JC	256FH	
2558	22	A1	49	SHLD	49A1H	
255B	CD	B0	39	CALL	39B0H	
255E	C2	6F	25	JNZ	256FH	
2561	CD	60	3C	CALL	3C60H	

Don't forget to

REN

2564	DA DD 21	JC 21DDH	TO BAD ARGUMENT
2567	7C	MOV A,H	
2568	B5	ORA L	
2569	CA EF 21	JZ 21EFH	
256C	22A3 49	SHLD 49A3H	
256F	CD 97 39	CALL 3997H	
2572	FE 0D	CPI CR	
2575	C2 DD 21	JNZ 21DDH	JUMP IF NOT CR TO BAD ARGUMENT
2578	11 FF FF	LXI D,0FFFFH	
257B	CD 69 3B	CALL 3B69H	
257E	2A AC 49	LHLD 49AC	(TOP OF BASIC DW)
2580	7E	MOV A,M	LOOK SEE INTO A
2581	FE 01	CPI 01	IS IT A 1
2583	CA B0 25	JZ 25B0H	
2586	CD 42 25	CALL 2542H	
2589	CD 9E 39	CALL 399EH	
258C	FE 0D	CPI CR	
258E	CA 80 25	JZ 2580H	
2591	CD 83 3C	CALL 3C83H	
2594	DA 89 25	JC 2589H	
2597	EB	XCHG	
2598	CD 69 3B	CALL 3B69H	
259B	DA 89 25	JC 2589H	
259E	C2 89 25	JNZ 2589H	
25A1	2A A5 49	LHLD 49A5	
25A4	EB	XCHG	

25A5 2A B4 48 LHL D 48B4H

25A8 2B DCX H

25A9 2B DCX H

25AA CD F8 36 CALL 36F8H

25AD C3 89 25 JMP 2589H

2583 → 25B0 2A A3 49 LHL D 49A3H

25B3 44 MOV B, H

25B4 4D MOV C, L } PUT HL INTO BC

25B5 2A A1 49 LHL D 49A1H

25B8 EB XCHG

25B9 2A AC 49 LHL D 49ACH (TOP OF BASIC DW)

25BC 7E MOV A, M

25BD FE 01 CPI 01

25BF C8 RZ RETURN IF a 1

25C0 E5 PUSH H

25C1 23 INX H

25C2 CD F8 36 CALL 36F8H

25C5 E1 POP H

25C6 CD 51 45 CALL 4551H

25C9 EB XCHG

25CA 09 DAD B

25CB EB XCHG

25CC C3 BC 25 JMP 25BCH

25CF 21 64 00 LXI H, 10RET

25D2 22 9E 48 SHLD 489EH

25D5 CD 95 3C CALL 3C95H

LIST

also on Typewriter
paper pages

25D8	E5	PUSH H	
25D9	CD B0 39	CALL 39B0H	
25DC	2A AA 48	LHLD 48AAH	
25DF	C2 EC 25	JNZ 25ECH	}
25E2	CD 95 3C	CALL 3C95H	
25E5	DA EC 25	JL 25ECH	
25E8	C2 EC 25	JZ 25ECH	
25EB	23	INX H	
25EC	22 3A 49	SHLD 493AH	
25EF	CD 97 39	CALL 3997H	
25F2	FE 0D	CPI CR	
25F4	C2 DD 21	JN5 21DDH to BAD ARGUMENT	
25F7	E1	POP H	
25F8	AF	XRA A	MAKE A=0
25F9	32 9D 48	STA 489DH	
25FC	7E	MOV A, m	
25FD	FE 01	CPI 1	
25FE	C8	RZ	
2600	11 3A 49	LXI D, 493AH	
2603	EB	XCHG	Return if parity even
2604	CD DB 39	CALL 39DBH	
2607	D0	RNC	Return if no carry
2608	EB	XCHG	
	11 C2 48	LXI D, 48C2	
	CD AF 24	CALL 24AF	looks at char + expands takes

LIST routine

INPUT
checked for line #'s after LIST
and checks their
legitimacy if
they are
present

25CF 21 64 00
22 9E 48
25D5 CD 95 3C
25D8 E5
CD B0 39
2A AA 48
25DF CZ EC 25
25E2 CD 95 3C
5 DA EC 25
8 CZ EC 25
B 23
25EC 22 3A 49
CD 97 39
FE 00
C2 00 21
→ 25F7 E1
25F8 AF
25F9 32 9D 48
25FC 7E
FE 01
C8 11 3A
11 3A 49
EB
CD DB 39
2607 D0
EB
11 C2 48
CD AF 24

LXI H, 10RET
SHLD 489E
CALL 3C95
PUSH H (4B01)
CALL 39B0 - on ps
LHLD 48AA
JNZ 25EC
CALL 3C95
JC 25EC
JNZ 25EC
INX H
SHLD 493A Δ
CALL 3997 - advance HL to next char, char into A
CPI CR - ~~check for line numbers~~
JNZ 21DD - BAD ARGUMENT ERROR
POP H - has 4B01 start of text
XRA A A=0
STA 489D
MOV A, M - *FIRST CHAR*
CPI 01 ARE WE AT THE END OF PROGRAM
RZ MARKER?
LXI D, 493A - ~~start of text~~
XCHG HL=493A/DE=###
CALL 39DB - see page 6 COMPARES DW add to by HL
with DE
RNC
XCHG HL=4B01/
LXI D, 48C2 - start of compressed input buffer
CALL 24AF - looks at chars and expands TOKENS

CALL 3997
CPI CR
LHLD 49AC - start of text ptr
RZ
CALL 3C60 - make LN binary
P into HL
JC 21DD - BAD ARG error
XCHG
CALL 3B69
JC 2201 - Ln error
RET

end of text DW ptr

23
E5
21 C2 48
CD C4 39
CD DB 3D
CD D6 3D
E1
261E C3 FC 25

INX H
PUSH H
LXI H, 48C2 *compressed input buffer*
CALL 39C4
CALL 30DB
CALL 30DB
POP H
JMP 25FC *on previous page*

*mvi C, CR
JMP 39CB - p. 6
- does some stuff + in the
and if JNPS WH1
if DB = CR*

~~25FC 7E
FE 01
C8
2600 11 3A 49
2603 EB
CD DB 39
D0
EB
11 C2 48
CD AF 24
23
ES
21 C2 48
CD C4 39
CD DB 3D
CD D6~~

~~MOV A, M
CPI 01
RZ
LXI D, 493A
XCHG
CALL 39DB
RNC
XCHG
LXI D, 48C2
CALL 24AF
INX H
PUSH H
LXI H, 48C2
CALL 39C4
CALL 30DB~~

Examples on SBL

4C29

26C9 BE 0D
32 A9 48

26CE AF
32 A8 48
CD 97 39

26D5 B7
F2 EF 26

26D9 FE 97
D2 D1 21

11 51 2E

entry point
from 21C0 if
a command

26E1 CD 9E 39
E6 1F

07

6F

26 00

19

CD E3 39

E9

we get here
if an assign-
ment
i.e. A=1+2

26EF CD 24 37

E5

F5

C5

06 F5

CD 8F 39

C1

F1

E1

26FD 3E 00

CA BD 36

2702 E5

CD 2E 2D

E1

C3 25 39

MVI A, CR
STA 48A9

XRA A

STA 48A8

CALL 3997

ORA A

JP 26EF

CPI 97

JNC 21D1

LXI D, 2E51

CALL 399E

ANI 1F

RLC (mult by 2)

MOV L, A

MVI H, 0

DAD D

CALL 39E3

PCHL

if not a token (there fore an assignment)

Syntax
ERROR

if A ≥ 97H

ANYTHING LESS THAN 97H
CAN BEGIN A LINE

point to vector table for non-CMD
-statements that can begin a line

point to correct vector

;LHLD(HL)

CALL 3724

PUSH H

PUSH PSW

PUSH B

MVI B, F5

CALL 398F

POP B

POP PSW

POP H

MVI A, 0

JZ 36BD

PUSH H

CALL 2D2E

POP H

JMP 3925

on return (for strings) (H) = start of string
DE = length BC points to len DW

token after variable name
must be a '='.

'=' token

CALL 399E/CMP B/RZ/JMP SYNTAXERROR

puts 0 into A without disturbing flags
if a string assignment statement
else it's a numeric assignment

Figure the result of the following expression?
get back destination address

PROPOSAL:

MOVE 2E3B thru 2FB8 inclusive

vector tables, Token lookup table
for statements that can begin a line

to where EDIT routine is now

4980-4C26 inclusive

and put EDIT at 2E3B thru 2FB8

WON'T FIT

PUT AUTO or etc Here?

or TAPE I/O cmds?

1100 0000
8421

#1 A=1
#2 B=2

#3 A=B

2193

ff
↓

LDA 48EA

CPI CR

JZ 2178

CALL 23EC compress input

JC 21AA

21AA

CALL 21B0

21B0

LXI H, 48C2

pt to compressed input

SHLD 48B4H

CALL 3997H

21B9

ANI EDH

AND first letter of input

CPI AD

LXI D, 2E3B

JZ 26E1

if ANDed A = 'AD'

CALL 26C9

26C9

3E 0D

32 A9 48

26CE

AF

32 A8 48

CD 97 39

26D5

B7

F2 EF 26

FE 97

D2 D1 21

11 51 2E

CD 9E 39

E6 1F

07

6F

26 00

19

CD E3 39

E9

MVI A, CR

STA 48A9

XRA A

STA 48A8

CALL 3997

ORA A

JP 26EF

next page

CPI 97

line number Header

JNC Syntax Error if A > 97H

LXI D, 2E51 start of non cond statements DW

CALL 399E

ANI 1F

RLC A = A * 2

find pos in table

MOV L, A

MVI H, 0

DAD D H = pos in table

CALL LALD(H, L)

PCHL

carry set when $\text{mem} > \text{DATA REG}$

2

26EF CD 24 37

CALL 3724

3724 CD 39 38

CALL 3839

3839 CD DC 38

CALL 38DC

38DC CD 97 39

CALL 3997

38DF FE 41

CPI 'A' (HL pts to 1st char)

D8

RC

FE 5B

CPI 'Z'+1

3F

CMC

38E5 D2 F3 38

JNC 38F3

23

INX

pt to next char

22 B4 48

SHLD 48B4

C9

RET

383C D8

RC

47

MOV B, A

0E 0F

MVI C, 0F

CD E9 38

CALL 38E9

38E9 CD 97 39

CALL 3997

EC FE 30

CPI '0'

points to '=' token

D8

~~RC~~

see if next char is numeric

FE 3A

CPI '9'+1

3F

CMC

D8 \ 23 \ 22B448 \ C9

RC \ INX \ SHLD 48B4 \ RET

3843 3F

CMC

00

RNC

3727 DA D1 21

SC Syntax error

372A CD 4F B8

CALL 384F

384F 3E 24

MVI 4, '#'

CD B2 39

CALL Check next char = A

3854 D8 RC if not a \$ (string)

372D D2 95 37 JNC 3795 if it was a string

3730 CD 97 39 CALL 3997
33 3E E0 MUL 4, E0

CD B2 B9 CALL 39B2

3738 CA 4A 37 JZ 374A

373B CD A0 38 CALL 38A0

38A0 78 MOV A, B put saved 1st char into A
D6 41 SUI 41H

38A4 87 ORA A set flags
2A AA 48 LHLD 48AA HL = top of test

23 INX H

CD 51 45 CALL 4551

4551 85 ADD L A = A + L
6F MOV L, A HL = HL + A
D0 RNC

38AB 54 MOV D, H
5D MOV E, L

CD E3 39 CALL LHLD(HL)

38BD 7C MOV A, H

B5 ORA L

CA BE 38 JZ 38BE

38B5 7E MOV A, M

23 INX H

B9 CMP C

C2 AB 38 JNZ 38AB

38BB 23 INX H

23 INX H

C9 RET

373E D2 8E 37 JNC 378E

378E F6 01 ORI 01

378E F6 01 ORI 01

378E F6 01 ORI 01

378E F6 01 ORI 01

378E F6 01 ORI 01

378E F6 01 ORI 01

378E F6 01 ORI 01

378E F6 01 ORI 01

'C' token
see if an array assign
if it was
(point to #?)

HL = top of test

A = A + L
HL = HL + A

see if HL = 0
means no data
has been initialized
get HBR byte
if not 0FH, then ending byte
= digit part of Varname
i.e. A3 B7 Z9
↑ ↑ ↑
point to first digit

3790 11 04 00
19
C9

LXI D, 04
DAD D
RET

point to sign

26F2 E5
F5
C5
06 F5
CD BF 39

PUSH H
PUSH PSW
PUSH B
MVI B, F5
CALL 398F

398F CD 9E 39
3992 B8
C8

CALL 399E now ptg to 'B'
CMP B Acc = 'B' token
RZ

26FA C1
F1
E1
3E 00
CA BD 36

POP B
POP PSW
POP H
MVI A, 0 (set A to zero w/o affecting flags)
JZ 36 BD

2702 E5
CD BE 2D

PUSH H
CALL 2D2E

cont on pg 6

2D2E 21 D1 2F
E5
21 22 F1
39
01 67 23
D2 20 22
AF
32 A7 48
2D40 3A A7 48
B7
C2 8A 2D
CD 39 38
2D4A DA 59 2D
CD 24 37

LXI H, 2FD1

PUSH H

LXI H, F122

DAD SP

SP was FF6 / now HL = 118

LXI B, 2367

JNC 2220

XRA A

STA 48A7

LDA 48A7

ORA A

JNZ 2D8A

CALL 3839

JC 2D59

CALL 372A

check for Complexity error

3rd on stack is AD points to loc of A's sign see prev pages

see prev pages

2050 CA F5 21
2053 CD 10 39

JZ 21FS
CALL 3910

(Type error)

3910 E5
11 FB FF
CD 33 39

PUSH H
LXI D, -5
CALL 3933

3933 2A 5F 49
E5
19

LHLD 495F
PUSH H
DAD D
CALL 3964

3938 CD 64 39

3964 D5
EB
21 B8 48
CD DB 39
396C DA 78 39
396F 21 5F 49
CD DB 39

PUSH D
XCHG
LXI H, 48B8
CALL 39DB
JC 3978
LXI H, 495F
CALL 39DB

compare DW ptr to
by HL with DE
HL=48B8, m DW=4C75
DE=5FF9

3975 EB
D1
D8

XCHG
POP D
RC

393B 22 5F 49
E1
C9

SHLD 495F
POP H
RET

3917 D1
EB
CD FB 38

POP D
XCHG
CALL 38FB

(pushed HL) (pts to B)
move var(HL) to var(DE)

38FB DE 05
7E
12
2B
1B

MVI C, 05
MOV A, M
STAX D
DCX H
DCX D

byte count of
PCD #
HL pts to 'B'
DE = work area
move var(HL) to
var(DE)

0D
C2 FD 38
C9

DCR C
JNZ 38FD
RET

if not done

391C AF
3C
32 A7 48
C9

XRA A } clear flags + set A to 1
INR A
STA 48A7
RET

2D56 C3 8A 2D
2D59

JMP 2D8A

2D8A C0 97 39
2D8D FE E1
D2 A0 2D
2D92 FE C0
D2 ED 2D
E1
3A A7 48
B7
CA D1 21
2D9F C9

CALL 3997 pt to CR
CPI E1
JNC 2DA0 if A < E1
CPI C0
JNC 2DED if A < C0
POP H now HL = 2FD1
LDA 48A7
ORA A
JZ 21D1 (Syntax error)
RET

2706 E1
C3 25 39

POP H new HL = (A)
JMP 3925

3925 EB
2A 5F 49
01 05 00
09
22 5F 49
C3 FB 38

XCHG now DE = A, HL = work (B)
LHLD 495F HL = same?
LXI B, 0005 both byte & word
DAD B
SHLD 495F
JMP 38FB

38FB 0E 05
7E
12

MVI
MOV
STAX

C 05
AM
D } move var

DCX H
DCX D
DCR C
JNZ 38FD
RET

21C6 CD 97 39
21C9 FE DD
C8

CALL 3997
CPI CR
RZ

21AD C3 5821

JMP 2158

"IF" also on following
Typewriter paper pages

IF

27F2 CD F4 39

27F5 CA 1A 28

CD 2E 2D

27FB CD 22 39

06 9D

2800 CD 8F 39

06 01

3A BE 48

2808 B7

CA 81 28

3E 9B

32 A9 48

CD 83 3C

D2 E7 28

C3 D2 26

281A 2A 5F 49

ES

ES

CD 26 36

ES

CD 9E 39

FE F7

D2 D1 21

FE EF

DA D1 21

CD 08 2E

23

CD E3 39

23

23

23

D1

C1

ES

CS

283E DS

CALL 39F4

JZ 281A

CALL 2D2E

CALL 3922

MVI B, THEN-Token

CALL 398F

MVI B, 1

LDA 48BE

ORA A

JZ 2881

MVI A, ELSE Token

STA 48A9

CALL 3C83

JNC 28E7

JMP 26D2

LHLD 495F

PUSH H

PUSH H

CALL 3626

PUSH H

CALL 399E

CPI NOT-token

JNC Syntax error if A > FB

CPI >= token

JC Syntax error

CALL 2E08

INX H

CALL 39E3

INX H

INX H

INX H

POP D

POP B

PUSH H

PUSH B

PUSH D

Complexity error checker
moves an FP #

goes to syntax error if what's in B #
what 399E gets

LHLD's HL

39F4 CD 97 39
EB
FE 90
CC 9E 39
CD DC 38
DA 09 3A
CD E9 38
D4 9E 39
EB
22 B4 48
FE 22
C8
FE 24
C8
FE 9A
C8
FE AB
C9

3A03

→ 3A09

CALL 3997
XCHG
CPI FN-token
CZ 399E
CALL 38DC
JC 3A09
CALL 38E9
CNC 399E
XCHG
SHLD 48B4
CPI "
RZ
CPI \$
RZ
CPI CHR\$
RZ
CPI old STR\$-token
RET

ALPHA + NUMERIC CHECKERS

38DC

CD 97 39

PE 41

D8

FE 5B

3F

D2 f3 38

C9

CALL 3997

CPI 'A'

RC

CPI 'Z'+1

CMC

JNC 38F3

RET

carry set if not A-Z

38E9

CD 97 39

PE 30

D8

FE 3A

3F

D8

23

22 B4 48

C9

CALL 3997

CPI '0'

RC

CPI '9'+1

CMC

RC

INX H

SHLD 48B4

RET

carry set if not 0-9

38F3

38F9

3922	21 BE 48	LAI H, 48BE
	EB	XCHG
	2A 5F 49	LHLD 49 5F
	01 05 00	LAI B, 5
	09	DAD B
	22 5F 49	SHLD 49 5F
	C3 FB 38	JMP 38FB

Top of memory scratch pad top ptr.

make room for result

MOVE FP # TO ANOTHER. SOURCE = HL DEST = DE

38F8	21 D8 34	LAI H, 34D8 - an FP 1
------	----------	-----------------------

→ 38FB	DE 05	MVI C, 5
--------	-------	----------

Byte count for 8 digit FP #

→ 38FD	7E	MOV A, m
--------	----	----------

	12	STAX D
--	----	--------

	2B	DCX H
--	----	-------

	1B	DCX D
--	----	-------

	00	DCR C
--	----	-------

	C2 FD 38	JNZ 38FD
--	----------	----------

3905	C9	RET
------	----	-----

IF

1001F01=20 THEN 0! &

000 2 3 3 2 9 2 9 0
EA 0 4 0 1 5 2 0 0 2 0

4801

27F2

CD F4 39

39F4 CD 97 39

39F7 EB

FE 90

CC 9E 39

39FD CD DC 38

38DC CD 97 39

38DF FE 41

D8

3A00 DA 09 3A

CD E9 38

D4 9E 39

3A09 EB

22 B4 48

FE 22

C8

3A10 FE 24

C8

FE 9A

C8

FE AB

C9

27F5 CA 1A 28

CD 2E 20

202E 21 01 2F

E5

21 22 F1

39

01 67 23

D2 20 22

AF

32 A7 48

3A A7 48

B7

C2 8A 20

2047 CD 39 38

3839 CD DC 38

383C D8

204A DA 59 20

CALL 39F4

CALL 3997

XCHG

CPI 'FN'

CZ 399E

CALL 38DC

CALL 3997

CPI 'A'

RC

JC 3A09

CALL 38E9

CNC 399E

XCHG

SHLD 48B4

CPI ''

RZ

CPI '\$'

RZ

CPI 'CHR\$'

RZ

CPI 'STR\$'

RET

JZ 281A

CALL 202E

LXI H, 2F01

PUSH H

LXI H, F122

DAO SP

LXI B, 2367

SP=FFA HC=11C

JNC 2220

'COMPLEXITY ERROR'

XRA A

STA 48A7

LDA 48A7

ORA A

JNZ 208A

CALL 3839

CALL 38DC

RC

JC 2059

neg PFDE?

2059 FE E5

CA A0 2D

CD 3A 3A

3A3A CD 97 39

3A30 EB

3A3E 21 BE 48

CD 59 43

4359 A5

32 76 49

32 75 49

32 77 49

3E F7

32 79 49

1A

FE 2B

CA 82 43

FE E3

CA 82 43

FE E5

CA 70 43

FE 2D

C2 83 43

→ 4382 1B

→ 4383 22 7A 49

3E 04

→ 2B

36 0

3D C2

C2 88 43

22 7C 49

EB

22 7E 49

CD 55 44

4455 2A 7E 49

7E

32 78 49

23

22 7E 49

FE 3A

D0

FE 30

3E

D0

CPI '-'

JZ 2DA0

CALL 3ABA

CALL 3997 ▽

XCHG ▲

LXI H, 480E

CALL 4359

XRA A

STA 4976

STA 4975

STA 4977

MVI A, 'NOT'

STA 4979

LDAX 0 char from

CPI '+'

JZ 4382

CPI '+' token

JZ 4382

CPI '-' token

JZ 437D

CPI '-'

JNZ 4383

INX H

SHLD 497A end of message

MVI A, 4 ————— byte count for message

DCX H

MVI M, 0

DCR A

JNZ 4388

SHLD 497C start of message

XCHG

SHLD 497E where we are in current line

CALL 4455

LHLD 497E

MOV A, M

STA 4978

INX

SHLD 497E

CPI :

RNC

CPI '0'

CMC

RNC

} see if numeric

- make ASCII string

EB 0E (ANI 0F) / 32 (SHR) / 01 RET

GOTO

28AB CD 83 3C

DA D1 21

CD 87 3A

EB

CD 98 3B

23

23

23

C3 E6 27

CALL 3C83

JC Syntax Error

CALL 3A87

XCHG

CALL 3B98

INT H

INT H

INT H

JMP 27E6

27E6

EB

CD 14 38

EB

22 B4 48

E1

C3 BA 26

XCHG

CALL 3814

XCHG

SHLD 48B4

POP H

JMP 26BA

current place in program

26BA

CD DB 3D

CD C9 26

CD 14 38

D2 BA 26

C3 9F 2A

CALL 3DD8

CALL 26C9

CALL 3814

JNC 26BA

JMP 2A9F

26C9

on previous pages

PRINT - 2AF9H

SP HL DE BC AF M
FFE 2AF9 2E51 9190 2416 CD

2AF9 CD 26 3A CALL 3A26
2AFC CD 97 39 CALL 3997
CR? 2AFF FE 00 CPI CR
2B01 CA D6 3D JZ 3D06
V? 2B04 FE 5C CPI \
2B06 CA D6 3D JZ 3D06
2B09 CD 97 39 CALL 3997
CR? 2B0C FE 00 CPI CR
2B0E C8 RZ
V? 2B0F FE 5C CPI \
2B11 C8 RZ
2B12 21 A9 48 LXI H, 48A9
2B15 BE CMP M
2B16 C8 RZ
TAB? 2B17 FE 9C CPI 'TAB' token
2B19 CA EB 2B JZ 2BEB
%? 2B1C FE 25 CPI %
2B1E CA 5C 2B JZ 2B5C
2B21 CD F4 39 CALL 39F4
2B24 CA 3F 2B JZ 2B3F

reference to next line

0011
4B06

9A92 9A → CHR token
9A86_m

9A02

9A92_m

9A86_m

9A02

00

9A86_m

9A83_{cm}

9A12

48A9

*
calls 3997 - CPI 'FN' token CPI ' ' CPI '\$' CPI 'CHR' token
38DC - CPI 'A' \ CPI 'E'
calls 3997

4B06

4B06

9A56_z

7BFE

FFC
does a lot of stuff - converts 50th of CHR(50) to ASCII equivalent

FFE 0001 7BFE 297F

E383_{cm} 32

7BFE 0001
HL = ten

0083_{cm}
0102

327F

0046

→ Print char in B

7BFD

0000

2B36 CD B0 39 CALL 3980
2B39 CA 09 2B JZ 2B06
2B3C C3 D6 3D JMP 3D06
2B3F 2A 5F 44 LHLD 495F
2B42 E5 PUSH H
2B43 CD 26 36 CALL 3626
2B46 D1 POP D
2B47 EB XCHG
2B48 22 5F 44 SHLD 495F
2B4B 7A MOV A, D
2B4C B3 ORA E
2B4D CA 36 2B JZ 2B36
2B50 46 MOV B, M
2B51 CD B9 3D CALL 30B9
2B54 CD DB 3D CALL 3DDB
2B57 2B DCX H
2B58 IB DCX D
2B59 C3 4B 2B JMP 2B4B

OUT

2CDD CD B8 3B

F5

3E D3

CD 12 2D

CD B0 39

CD B8 3B

2CEC

47

F1

FE 11

DA F7 2C

78

CB 54 0C

2CF7

CALL

3BB8

get port #

PUSH

PSW

save port #

MVI

A,

OUT opcode

CALL

2D12

port (OUT port #/RET) into C54

CALL

39B0

check for comma

CALL

3BB8

get data

MOV

B, A

POP

PSW

get back port #

CPI

11

JC

2CF7

if port < 11H

MOV

A, B

JMP

0C54

and RET to 21C6 if direct mode

or 26C0 if running

puts OUT/IN code + RET into C54-C56

2D12 21 54 0C

77

7A

B7

C2 EF 21

23

73

23

36 C9

C9

LXI H, 0C54

MOV M, A

MOV A, D high byte of port # DW

ORA A

JNZ 21EF too big # error?

INX H

MOV M, E put port # into mem

INX H

MVI M, C9 RET opcode

RET

Conditional tester

3044	F5		PUSH PSW
	C5	1A 67	PUSH B
	1A		LDAX D
	B7		ORA A
3048	F2	55 30	JP 3055
	0A		LDAX B
	B7		ORA A
	F2	55 30	JP 3055
	EB		XCHG
3051	50		MOV D, B
	59		MOV E, C
	44		MOV B, H
	4D		MOV C, L
3055	1A		LDAX D
	EE	80	XRI 80
	67		MOV H, A
	0A		LDAX B
	EE	80	XRI 80
	BC		CMP H
305D	C2	7A 30	JNZ 307A
	FE	80	CPI 80
	CA	7A 30	JZ 307A
	Z1	FC FF	LXI H, FFFC (or -4)
	E5		PUSH H
	EB		XCHG
	19		DAD D
	EB		XCHG
306C	09		DAD B
	C1		POP B
	EB		XCHG
306F	1A		LDAX D
	BE		CMP m
3071	C2	7A 30	JNZ 307A
	13		INX D
	23		INX H
	0C		INR C
	C2	6F 30	JNZ 306F

get EXP, sign, of #
complement Bit 7

and put into H
get EXP, sign of #
" " "

and compare with H
if not equal exponents or signs

BC = FFFC
HL = 1st #, DE = 2nd # to be compared with
exchange pointers

compare the two #'s first bytes

that byte was equal, so
increment to next byte
and increment byte counter
if not done

307A DI
 EI
 F5
 D5
 24
 307F F4 f838
 E1
 F1
 3E 00
 C9

POP D
 POP H
 PUSH PSW
 PUSH D
 INR H
 CP 38F8
 POP H
 POP PSW
 MVI A, 0
 RET

gets pushed DE
 puts a 0 into A, but doesn't set zero flag

38F8 21 D8 34
 0E 05
 38FD 7E
 12
 2B
 1B
 0D
 C2 F0 38
 C9

LXI H, 34D8
 MVI C, 05
 MOV A, m
 STAX D
 DCX H
 DCX D
 DCR C
 JNZ 38FD
 RET

Move a FP BCD '1' to DE

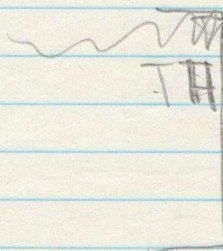
ASC
3189 =

LEN
317C

3189 CD 85 39 CALL 3985
318C CD 24 37 CALL 3724
318F C2 F5 21 JNZ Type error
3192 7A MOV A, D
3193 B3 ORA E
3194 CA EF 21 JZ Out of bounds
6E MOV L, m
3198 26 00 MVI H, 0
319A C3 D5 30 JMP 3005

317C CD 85 39 CALL 3985
317F CD 24 37 CALL 3724
3182 C2 F5 21 JNZ Type error
3185 EB XCHG
3186 C3 D5 30 JMP 3005
→ stores HL as BCD FP?

3724 - points HL to 1st char of string or exponent of number
DE = length of string (or number?)
ZERO flag set if a string
3985 - makes sure character after ASC or LEN = "C"



LDAX D
MOV M, A
CALL next
INX D

4C95 = A0

BART
BART

3993 08
317F CD
3182 C2

T
CALL T#1

check that next char is a comma (when done, carry set if not)
 call 39B2 with A already loaded with a char

39B0 3E 2C

39B2 F5

CD 9E 39

6 F1

2B

BE

C8

37

C3 35 38

39BE 19

0D

C2 AE 39

C9 check

mvi A, '9'

PUSH PSW

CALL get next char

POP PSW

DCX H

CMP M

RZ

STC

JMP 3835

DAD D

DCR C

JNZ 39BE

RET

39C4 0E 0D
 C3 CB 39
 39C9 0E 22
 39CB 7E
 CE 47
 CD B9
 CE C8
 CF FE 0D
 D7 CA D1 21
 D4 CD B9 3D
 D6 23
 D8 C3 CB 39

39DB

3DB9 CD ED 20
 3DBC FE 0D
 CA CC 3D
 FE 20
 D8
 3A A6 48

MVI C, CR
 JMP 39CB
 MVI C, ""
 MOV A, m
 MOV B, A
 CMP C
 RZ
 CPI CR
 JZ 2101
 CALL 3DB9
 INX H
 JMP 39CB

CALL 20ED
 CPI CR
 JZ 3DCC
 CPI 'x'
 RC

prints character in B, moves it into A *checks for CR*

return if less than 20H

TIME function (and)

Get a param into DE and A

DE has argument

30DE CD BB 3B

CALL 3BBBH

get # in param into DE

30E1 2A 00 0C

LHLD TIMER (0C00H)

get TIME

30E4 AF

XRA A

make A=0

30E5 BA

CMP D

look at D

30E6 C2 A2 3B

JNZ 3BA2H

if not zero, ^{JUST LOOKING} AT TIME

30E9 BB

CMP E

look at E

30EA C2 A2 3B

JNZ 3BA2H

if not zero, " "

30ED EB

XCHG

put DE (all zeroes) into H

30EE 22 00 0C

SHLD TIMER

Zero timer

30F1 EB

XCHG

put 'em back where it was

30F2 C3 A2 3B

JMP 3BA2H

exit

3BB8 CD 2E 2D

CALL 2D2E

3BBB CD 06 39

CALL 3906H

3BBE E5

PUSH H

save HL

3BBF CD 9D 31

CALL 319DH

3BC2 CD 22 39

CALL 3922H

3BC5 E1

POP H

restore HL

3BC6 7E

MOV A, M

3BC7 11 FB FF

LXI D, 0FFFFBH

3BCA 19

DAD D subtract 5 from HL

3BCB 11 00 00

LXI D, 0

3BCE 0RA A

set flags

3BCF C8

RZ

3BD0 FA EF 21

JM 21EFH

! "Out of Bounds error if sign flag set

3BD3 EE 00

XRI 0C0H

3BD5	4F	MOV C,A	
3BD6	0D	DCR C	
3BD7	23	INX H	
3BD8	7E	MOV A,M	
3BD9	0F	RRC	
3BDA	0F	RRC	
3BDB	0F	RRC	
3BDC	0F	RRC	
3BDD	CD F5 3B	CALL 3BF5H	DE = DE * 10 and other junk w/ A
3BE0	DA EF 21	JC 21EFH	!"Out of bounds error" if carry set
3BE3	0D	DCR C	
3BE4	F2 F2 3B	JP 3BF2H	JMP IF UNDER 80H
3BE7	7E	MOV A,M	
3BE8	CD F5 3B	CALL 3BF5H	APPARENTLY MAKES DE = DE * 10 and STUFF WITH A
3BEB	DA EF 21	JC 21EFH	!"Out of bounds error" if carry set
3BEE	0D	DCR C	
3BEF	FA D7 3B	JM 3BD7H	
3BF2	7A	MOV A,P	
3BF3	B3	ORA E	if 0=0, then A=E
3BF4	C9	RET	

3BF5	E5	PUSH H	SAVE HL
3BF6	62	MOV H,D	} PUT DE INTO HL
3BF7	6B	MOV L,E	
3BF8	29	DAD H	DE = HL + HL = 2 * HL
3BF9	DA 11 3C	JC 3C11H	on next page

3BFC 29

3BFD DA 11 3C

3C00 19

3C01 DA 11 3C

3C04 29

3C05 DA 11 3C

3C08 EB

3C09 E6 0F

3C0B 83

3C0C 5F

3C0D 7A

3C0E CE 00

3C10 57

3C11 E1

3C12 C9

DAD H $2HL + 2HL = 4*HL$

JC 3C11H

DAD D $HL = 4*HL + DE = 5*HL$

JC 3C11H

DAD H $HL = 5*HL + 5*HL = 10*HL$

JC 3C11H

XCHG put 10*original DE into DE

ANI 0FH MAKE 4MSB's of A=0

ADD E $A = E + A$

MOV E, A PUT A into E

MOV A, D PUT D into A

ACI 0 add just carry to A

MOV D, A put new A into D

POP H

RET

3BA2 11 3A 49

3BA5 D5

3BA6 CD AC 3C

3BA9 3E 00

3BAB 12

3BAC D1

3BAD 21 BE 48

3BB0 E5

3BB1 CD 59 43

3BB4 E1

3BB5 C3 10 39

LXI D, 493AH

PUSH D

CALL 3CACH

MVI A, 00H = 0

STAX D

POP D

LXI H, 48BEH

PUSH H

CALL 4359H

POP H

JMP 3910

3C60 - CALL 3997

3C63 - SHLD 497E

3C66 - CALL 4455

4455 LHL D 497E
MOV A, M
STA 4978
INX H
SHLD 497E
FE ':'
RNC
PC '0'
CMC
RNC
ANI 0FH
STC
RET

check if
number
for LN's

sets carry flag if DW 497E points to a number
(ASCII 30-39)

3C69 - CMC

- RC

LXI D, 0

→ 3C6E - CALL 3BFS - multiplies DE by 10 and ANI 0FH

JC 21EF - out of bounds error

CALL 4455 - above

JC 3C6E - gets next LN

3C7A LHL D 497E

DCX H

SHLD 48B4

XCHG

RET

3C60 CALL 3997

63 SHLD 497E

66 CALL 4455

69 CMC

6A RC

6B LXI D,0

6E CALL 3BF5

JC 21EF

CALL 4455

JC 3C6E

SHLD 497E

DCX H

SHLD 48B4

XCHG

RET

see p. 2 make LW binary?

3BF5 PUSH H

MOV H,D

MOV L,E

DAD H

3BF9 JC 3C11

DAD H

JC 3C11

DAD D

JC 3C11

3C04 DAD H

JC 3C11

3C08 XCHG

3C09 ANI 0F

3C0B ADD E

3C0C MOV E,A

3C0D MOV A,D

- PUT DE INTO HL

- HL = $\square * 2$

HL = $\square * 4$

HL = $\square * 5$

HL = $\square * 10$

ACI 0

3C10 MOV D,A

3C11 POP H

- mask off upper nybble of A

3CAC AF

XRA A

LXI B, D8F0

CALL 3CD1

3CB0

3CB3

LXI B, FC18

3CB6

CALL 3CD1

3CB9

LXI B, FF9C

3CBC

CALL 3CD1

3CBF

LXI B, FFF6

3CC2

CALL 3CD1

3CC5

LXI B, FFFF

3CC8

CALL 3CD1

3CCB

RNZ

3CA1=DA
3CAB=C9

3CD1 PUSH D

3CD2 MVI D, FF

3CD4 SHLD 48A4

INR D

D=00

3CD8 DAD B

JC 3CD4

LHLD 48A4 — FIRST LN

MOV B, D

POP D

DRA B

RZ

3CE3 — MVI A, 30

ADD B

STAX D

RET

Get + echo input for commands + others (i.e. 'INPUT')
 at end HL pts to last char / A = last char

(1559318)

3CE9	21 EA 48	LXI	H, 48EA	(start of uncompressed input buffer)
	3A A7 49	LDA	49A7	
	B7	ORA	A	
	CA A7 3D	JZ	3DA7	if (49A7) = 0 goto startup ^{msg} handler
3CF3	11 83 33	LXI	D, 3383	?
	01 50 01	LXI	B, 0150	?
3CF9	CD 05 3D	CALL	3D05	input + echo into buffer rtn
	FE 0D	CPI	CR	
	C8	RZ		if line ended in CR
	01 A0 23	LXI	B, 23A0	else error
	C3 20 22	JMP	2220	goto length error
3D05	see next page			

2100 304F 307A 3088 3034

WH0 + WH1 inputting routine

3005	D5	PUSH D	
6	ES	PUSH H	- has where to put chars
7	C5	PUSH B	?C has max # chars?
8	06 00	MVI B, 0	- char. ctr
300A	CD BA 20	CALL BASIC's wh0	
	FE 7F	CPI R0out	
	CA 4C 3D	JZ 304C	
	FE 18	CPI CAN	
	CA 78 3D	JZ 3078	
	FE 17	CPI Worderase	
	CA 83 3D	JZ 3083	
	77	MOV M, A	
	23	INX H	
	04	INR B	
	FE 00	CPI CR	
	CA 3E 3D	JZ 303E	
3024	FE 1B	CPI ESC	
3026	CA 3E 3D	JZ 303E	
	CD 24 0C	CALL WH1	
	FE 09	CPI HTab	
	C2 39 3D	JNZ 3039	
3031	E5	PUSH H	
	2A 0E 0C	LHLD 0C0E	cursor position
	2B	DCX H	
	36 3F	MVI M, 03FH	

304F
307A
3088

1089
- 28 char
1071

;TAB rtn

3D38	E1	POP H	
3D39	78	MOV A, B	←
	B9	CMP C	
	C2 0A 3D	JNZ 3D0A	↗
3D3E	58	MOV E, B	; go here for RET or ESC
	C1	POP B	
	78	MOV A, B	
	B7	ORA A	set flags
	2B	DCX H	
	7E	MOV A, M	
3D44	CC 24 0C	CZ WH1	
	7 43	MOV B, E	
	E3	XTHL	
	E1	POP H	
	D1	POP D	
	C9	RET	
3D4C	CD 52 3D	CALL 3D52	; RUBOUT rtn
3D4F	C3 0A 3D	JMP 3D0A	
→ 3D52	78	MOV A, B	; Also RUBOUT rtn
	B7	ORA A	
	C8	RZ	
3D55	05	DCR B	decr char chr
	2B	DCX H	decr buffer ptr
	7E	MOV A, M	get deleted char
	FE 09	CPI HTAB	was it a tab
3D5A	CA 60 3D	JZ 3D60	

3050 FE 20

5F D8

60 E5

→ 3061 3E 7F

CD 24 0C

2A 0E 0C

3069 7D

E6 3F

CA 76 3D

2B

7E

FE 7F

3073 CA 61 3D

→ 3076 E1

C9

3078 78

9 B7

307A CA 0A 3D

CD 52 3D

E3 78 3D

3083 CD 52 3D

78

B7

3088 CA 0A 3D

2B

7E

CPI 'B'

RC

return if > 'B'

PUSH H

MVI A, DEL

DISPLAY a backspace (CALL WHI)

LHLD 000E (CURPOS)

MOV A, L

ANI '3F'

JZ 3076

DCX H

MOV A, m

CPI 'DEC'

JZ 3061

POP H

RET

MOV A, B ; CAN Rtn

ORA A

JZ 300A

CALL 3052 (rubout routine)

JMP 3078 repeat until a 0

CALL 3052 (rubout rtn) ; WORD ERASE Rtn

MOV A, B

ORA A

JZ 300A

DCX H

MOV A, m

3D8D 23

FE 30

DA 0A 3D

FE 3A

DA 83 3D

3D98

EB 5F

FE 41

DA 0A 3D

FE 5B

DA 83 3D

C3 0A 3D

INX H

CPI '0'

JC 3D0A

CPI '9'+1

JC 3D83

ANI '5F'

CPI 'A'

JC 3D0A

CPI 'Z'+1

JC 3D83

JMP 3D0A

if A > '0'

repeat if A ≥ '9'

if A > 'A'

repeat if A ≥ 'Z'+1

-done-

3DA7

see next page

Start up message handler

3DA7	3E 80	MVI	A, 80	
9	32 A7 49	STA	49A7	
C	11 0A 20	LXI	D, 200A	(1st byte of startup message)
→ 3DAF	1A	LDAX	D	
	77	MOV	m, A	
	13	INX	D	
	23	INX	H	
	FE 0D	CPI	CR	
	C8	RZ		
	C2 AF 3D	JNZ	3DAF	(why JNZ? why not JMP?)
3DB9	CD ED 20	CALL	20ED	if B=CAN, call can rtn, else print char also, puts B into A
	FE 0D	CPI	CR	
	CA CC 3D	JZ	3DCC	
	FE 20	CPI	'0'	
	D8	RC		return if < '0'
	3A A6 48	LDA	48A6	
	3C	INR	A	
	32 A6 48	STA	48A6	
	C9	RET		
3DCC	AF	XRA	A	
	32 A6 48	STA	48A6	
	C9	RET		
3DD1	3A A6 48	LDA	48A6	
	B7	ORA	A	
	C8	RZ		
3DD6	06 00	MVI	B, 00	

3DD8 C3 B9 3D

JMP 3DB9 on prev page

3F31 C6 01
 3F33 IF
 FS
 11 90 49
 0E 04
 AF
 12
 3F3C 1B
 2B
 7E
 12
 3F40 0D
 3F41 C2 3C 3F
 44 FI
 FS
 21 90 49
 3F49 B7
 CA 56 3F
 4D 4F
 AF
 4F 2B
 1B
 3F51 12
 0D
 C2 4F 3F
 3F56 FI

ADI 01
 RAR
 PUSH PSW
 LXI D, 4990
 MVI C, 4
 XRA A
 STAX D
 DCX D
 DCX H
 MOV A, M
 STAX D
 DCR C
 JNZ 3F3C
 POP PSW
 PUSH PSW
 LXI H, 4990
 ORA A
 JZ 3F56
 MOV C, A
 XRA A
 DCX H
 DCX D
 STAX D
 DCR C
 JNZ 3F4F
 POP PSW

set flags

3F57 DA 62 3F

3F5A 23

54

5D

CD 88 3E

1B

EB

3F62 7E

FE 5D

2B

E5

D4 BB 3E

D1

F1

E1

AE

3F6E 46

2B

C5

F2 E8 3F

JC 3F62

INX H

MOV D, H

MOV E, L

CALL 3E88

DCX D

XCHG

MOV A, m4

CPI 'P'

DCX H

PUSH H

CNC 3EBB

POP D

POP PSW

POP H

XRA m

MOV B, m

DCX H

PUSH B

JP 3FE8

3FEB

D5

06 04

B7

3FEC

1A

8E

27

12

2B

1B

05

C2 EC 3F

3F66

DA 03 40

E1

F1

D1

12

1B

0E 04

C3 FD 38

PUSH D

MVI B, 4

#byte count

ORA A

LDAX D

ADC M

DAA

STAX D

DCX H

DCX D

DCR B

JNZ 3FEC

JC 4003

POP H

POP PSW

POP D

STAX D

DCX D

MVI C, 04

JMP 38FD

128 64 32 16 8 4 2 1

SIGN SIGN
#? EXP?

EXP

449B 2A C0 48

9E EB

9F 2AAC 49

A2 7E

FE 01

CA BE 44

23

CD DB 39

44AC 2B

0E 02

~~5A~~ 03 44

44B2 0E 01

DA D3 44

7E

CD 51 45

C3 A2 44

44BE 3A BF 48

FE 04

C8

CD 56 45

2A AA 48

CD 28 45

36 01

22 AA 48

C9

44D3 46

LHLD 48C0 "Input Line's" number

XCHG

LHLD 49AC start of text ptr

MOV A,m

CPI end of program char

JZ 44BE

INX H

CALL 39DB

DCX H

MVI C,2

JZ 44D3

MVI C,1

JC 44D3

MOV A,m

CALL 4551

JMP 44A2

LDA 48BF

CPI length of 4

RZ

CALL 4556

LHLD End of text ptr

CALL 4528

MVI m,01

SHLD 48AA

RET

MOV B,m

line length of input line

4404 22 73 49
 3A BF 48
 0D
 440B CA EC 44
 D6 04
 CA E5 44
 44E3 C6 04
 44E5 90
 CA 1F 45
 DA 0B 45
 44EC 47
 3A BF 48
 FE 04
 44F2 08
 78
 CD 56 45
 44F7 2A 73 49
 CD 45 45
 2A AA 48
 EB
 22 AA 48
 03
 CD 3F 45
 C3 1F 45
 450B 2F
 3C

SHLD 4973 4
 LDA 48BF ^{input line's}length
 DCR C
 JZ 44EC
 SUI 04
 JZ 44E5
 ADI 04
 SUB B
 JZ 451F
 JC 450B
 MOV B, A
 LDA 48BF ^{input line's}length
 CPI length of 4
 RZ
 MOV A, B
 CALL 4556
 LHLD 4973 4
 CALL 4545
 LHLD 48AA ^{end of text ptr.}
 XCHG
 SHLD 48AA " " "
 INX B
 CALL 453F
 JMP 451F
 CMA
 INR A

449B LHL D 48C0
XCHG
LHL D 49AC - start of test pointer

44A2 MOV A, m
44A4 CPI 01 - if no program

44A7 JZ 44BE

44A8 INX H

44A9 CALL 39DB
DCX H
MVI C, 2

44AF JZ 44D3

MVI C, 1

JZ 44D3

44B7 MOV A, m

CALL 4551

44BB JMP 44A2

44BE LDA 48BF

CPI 4

RZ

CALL 4556

LHL D 48AA - top of test pointer

CALL 4528

MVI M, 1

SHLD 48AA

RET

44D3

4455 LHL D 497C
MOV A, m
STA 4478
INX H
SHLD 497E

FE ':'

RNC

FE '0'

CNC

RNC

ANI 0F

STC

RET

check
if memory
(LN's)

4567	CD 3D 48	CALL TAPE IN?	45A4	06/AA/40/CE00	SETUP DATA for BYTE
C9		RET	C9		RETURN
4568	CD 2D 48	CALL TAPE OUT	45AA	CALL 45B5	(CD B5 45)
C9		RET	11834.10	05/AA/40/0C/E6/E6/D	DATA FOR POLY
* T1SR IN and OUT *			C9		RETURN
456F	DB 01	IN 1	45B5	F3	Disable Int.
IF		RAR	AF		XRA A
DA 8545	JC 4585		D3 01		OUT 1
IF		RAR	216F45		LXI H, 456F
DZ 6400	JNC 064	(IORET)	22160C		SHLD 0C16 (SRAY)
DB 00	IN 0		45BF	216745	LXI H, 4567
21 0A0C	LXI H, 0A	RBUF	11280C		LXI D, 028 (WHZ)
36 00	MVI m, 0		45C5	06 08	MVI B, 8
23	INX H		45C7	7E	MOV A, m
77	MOV m, A		45C8	12	STAX D
C3 6400	JMP IORET		23		INX H
21 080C	LXI H, 08	TBUF	45CA	13	INX D
36 00	MVI m, 0		05		DCR B
23	INX H		45CC	C2 C7 45	JNZ 45C7
7E	MOV A, m		FB		Enable Int
D3 0000	OUT 0		C3 AD 02		JMP 2AD (SETUP)
458E	C3 6400	JMP 064 (IORET)	45D3	AF	XRA A
4591	3A AA49	LDA 49AA	47		MOV B, A
B7		ORA A	CD 280C		CALL WHZ
C2 9B45	JNZ 459B		F5		PUSH PSW
4598	CD 1825	CALL 2518	3A AA49		LDA 49AA
459B	AF	XRA A	B7		ORA A
D3 01		OUT 1	CA EA 45		JZ 45EA
C3 5821	JMP 2158		FI		POP PSW
45AI	CD B5 45	CALL 45B5	EB		XCHG
			BE		CMP m
			45E3	EB	XCHG

15A8T SETUP

SETUP

of verifying

OVER

4611 CD 97 39
4614 2A B4 48
4617 0E 09

CALL 3997H
LHLD 48B4H ? correct program pointer?
MVI C, 9

4619 7E MOV A, m

461A 23 INX H

461B FE 80 CPI 80H

461D D2 56 46 JNC 4656H

4620 FE 20 CPI '0'

4622 CA 38 46 JZ 4638H

4625 FE 2C CPI ','

4627 CA 38 46 JZ 4638H

462A FE 0D CPI CR

462C CA 38 46 JZ 4638H

462F 0D DCR C

4630 CA FF 3C JZ 3CFFH

4633 12 STAX D

4634 13 INX H

4635 C3 19 46 JMP 4619H

4638 22 B4 48 SHLD 48B4H

463B 47 MOV B, A

463C 79 MOV A, C

463D FE 01 CPI 01

463F CA 4A 46 JZ 464AH

4642 3E 20 MVI A, '0'

4644 12 STAX D

4645 13 INX H

4646 0D DCR C

4647 C3 3C 46 JMP 463CH

464A

48C3
AS
load 2C

expand the token

GET FILENAME
+ PUT IT AT
0C54H on

exit

01 40 23 LXI B, 23AH
C3 20 22 JMP 2220H
CANCEL RTN

length error
message

next page

put blanks
until 0C54H on
= eight char

TAPE

see also tape page

464A	78	MOV	A, B
464B	FE 2C	CPI	' , '
464D	CA 76 46	JZ	467CH
4650	3A 08 20	LDA	Tape Mode ^{Default} Char.
4653	C3 7F 46	JMP	467FH

TAPE

if no comma, add in default

4656	EB	PUSH	A
4657	21 7F 2E	LXI	H, 2E7F

4656 thru 467B
on next page

467C	CD 9E 39	CALL	399EH
467F	FE 42	CPI	'B' byte
4681	CA A1 45	JZ	45A1H
4684	FE 50	CPI	'P'
4686	CA AA 45	JZ	45AAH
4689	C3 D1 21	JMP	21D1H syntax error

A has taken to be found & expanded
 changes taken in a filename to the reserved word

4656	E5		PUSH H	
4657	21	7F 2E	LXI H, 2E7FH	- start of STMT table
465A	47		MOV B, A	put first token into B
465B	7E		MOV A, M	
465C	B8		CMP B	
465D	CA	68 46	JZ 4668H	Jump if match
4660	3C		INR A	FF is end of table, so increment
4661	23		INX H	
4662	C2	5B 46	JNZ 465BH	if A was FF, then it is 0
4665	C3	D1 21	JMP 2101H	EXIT TO "SYNTAX ERROR"
4668	23		INX H	
4669	7E		MOV A, M	
466A	FE	7F	CPI 7FH	
466C	D2	78 46	JNC 4678H	(JUMP IF > 7FH) } why not 0x7A or 7m 4678
466F	12		STAX D	
4670	13		INX D	
4671	0D		DCR C	
4672	CA	FF 3C	JZ 3CFFH	EXIT
4675	C3	68 46	JMP 4668H	
4678	E1		POP H	
4679	C3	19 46	JMP 4619H	

24 0C WHI 20F4 / 302A / 3045 / 3064 / 46C8 / 4829
 20 0C WHI ~~20F4~~
 0A 20 BASICs WHI 2057 / 3202 / 300B /
 64 00 10RET 2051 / 2092 / 2098 / 220C / 2309 / 2328 / 2350 / 2365 / 2399 /
 2500 / 2633 / 2648 / 2776 / 267C / 2E34 / 3663 / 3939 / 3958 /
 4089 / 4583 / 458F / 4710 / 4577 / 4809 / 489E
 00 0C TIMER 30E2 / 30EF

WIRE UP AS 25
 WIRE IN 74367 to check STATE
 of 8212

2708 CALL CLEAR

ET

! Write or Read Or Timeshare

if W Go to SIM on 2708

if R return

if T goto Timeshar

81 ADDC
4820 JMP 4824? JUMPED
4824 F5 PUSH PSW
78 MOV A, B
C6 30 ADI 30 MAKE # ASCII
C0 24 00 CALL WH1
F1 POP PSW
C9 RET

456F = TISK in + out

4567 = C03D 48/C9
456B = C02D 48/C9

BASIC'S WH3

-184710
4820 E5 PUSH H
482E 21 08 0C LXI H, C08 (TRUTH AND)
4831 F5 PUSH PSW
4832 7E MOV A, m^{m=02} HL=C08
4833 B7 ORA Aⁿ⁼⁰¹
4834 C2 32 48 JNZ 4832
4837 34 INR m^{HL=C08}
38 23 INX H^{HL=C09}
39 F1 POP PSW
3A 77 MOV m, A
BB E1 POP H
3C C9 RETURN

-1849310
4830 E5 PUSH H
483E 21 0A 0C LXI H, C0A (R.B.V.F.)
↓ ↓ ↓
4841 7E MOV A, m
4842 B7 ORA A
4844 C2 41 48 JNZ 4841
4846 35 DCR m
4847 23 INX H
↓ ↓ ↓
4848 7E MOV A, m
4849 E1 POP H
484A C9 RET

ALMOST EXACTLY LIKE DUMP SMD

484B 3E 21 MVI A, 21H
03 01 OUT 1
3A 74 0C LDA 0C74
4852 C0 08 48 CALL 4808
4855 C0 D6 30 CALL 3D06^{OUTPUTS CR}
4858 21 FF 8F LXI H, 8FFF
4808 0E 64 MVI C, 64
CD 16 48 CALL 4816
0E 04 MVI C, 04
CD 16 48 CALL 4816
4812 47 MOV B, A
4813 C3 24 48 JMP 4824
4816 06 00 MVI B, 0
4816 91 SUB C
4819 FA 20 48 JM 4820
481D C3 18 48 JM 4818

47F2 06 0 MVI B, 0
4F MOV C, A
47F5 7E MOV A, m
23 INX H
F5 PUSH PSW
80 ADD B
47 MOV B, A
F1 POP PSW
CD 2C 0C CALL 0C2C (WH3)
47FE 0D DCR C
47FF C2 F5 47 JNZ 47F5
278 2F MOV A, B
3C CMA
5 3C INR A
5 C3 2C 0C JMP WH3

THIS IS THE SMD

4808

C74 = WriteTape filename (WNAME)

C76 = WLEN

C77^{C78} = WADR

C74 =

20A6

337B₁₆ - 3397₁₆ > 50
1379₁₀ - 13207₁₀

BASIC MEMORY MAP

HEX ADDRESS	DECIMAL ADDRESS	FUNCTION
^{CCF} CCF (after C60)	3197-3266	unflipped top down input buffer
2000	8192	COLD START ADDRESS
2003	8195	WARM START ADDRESS
2006	8198	FRONT PANEL ENTRY CODE (CTL-Z)
2007	8199	INTERRUPT BASIC CODE (ESC)
2008	8200	TAPE MORE CHARACTER "B" or "P"
2009	8201....	prompt character
200E	8206	BASIC version text
204B 20ED	8285-8286?	CANCEL ROUTINE OUTPUT ROUTINE
205D-205E	8285-8286?	End of Basic address (49B0)
2060-2061	8288-8289?	end of memory search start address (49FF)
2069	8298?	start of kybrd interrupt setup
20BA		WH0 kybrd entry point
22D0 2353	8911-9195	error messages
	11447-11464	'Input error - retype'
3005 return addr for CALLs	11796-11830	'error' 'in line' 'Stop' 'Interrupted' ''
3006 - CR RTN 4808	11903-12217	Command lookup table
482D	?18477?	TAPE - DISPLAY RCD # WH3 routine (OUT) found at 0C2CH
483D	?18544?	WH2 routine (tape in) go to 0078H
48A0 48A2? 48C3 48EA	18627-18664	Flipped input (approx size - uses also)?
	18665-18745	Converted input (Cnvgd REN to n, etc) (approx size - uses also)?
49B0	18864	Start of programs (usually)
47F2	***	outputs 1 record (LIKE PUT IN SMD)
C74 - # of records (write)		something for tape
49AA - USED OFTEN		buffer or variable area
4920 48AA -		
493A		
484B	***	ALMOST EXACTLY LIKE DUMP IN SMD

BASIC'S TAPE DUMPER

WADR, etc on back page

C77 is referenced at 477AH, 47D0H, 47D7H, & 4870H
 ↳ SHLD 0C77H ↳ LHLD 0C77H ↳ SHLD 0C77H ↳ LHLD 0C77H

-1850710

```

484B 3E 21 DUMP: MVI A, 021H }
484D 03 01 OUT 1H } TURN ON MOTOR+USART
484F 3A 74 0C ≠ LDA WRCD - LOAD A with RCD # which is 2
4852 CD 08 48 CALL 4808 - DISPLAY IT AS RECORD #
4855 CD D6 3D CALL 3DD6 - output a CR
4858 21 FF 8F LXI H, 8FFFFH
485B 2B DELAY: DCX H
485C 7C MOV A, H
485D B7 ORA A
485E C2 5B 48 JNZ DELAY
4861 0E 40 MVI C, 40H or 6410
4863 3E E6 DUMP0: MVI A, 0E6H - sync character *
4865 CD 2C 0C * CALL WH3
4868 0D DCR C
4869 C2 63 48 JNZ DUMP0
486C 3E 01 MVI A, 01 - start of header
486E CD 2C 0C CALL WH3
4871 3E 0E MVI A, 0EH14 - length of header record
4873 21 6C 0C LXI H, WNAME(0C0C) - file name
4876 CD F2 47 CALL PUT - outputs header
4879 3A 76 0C LDA WLEN - length of record
487C 2A 77 0C LHLD WADR - address WADR
487F EB XCHG
4880 2A 7A 0C LHLD 0C7AH BIAS ADDR
4883 19 DAD D
    
```

↑
EXACTLY
LIKE
DUMPER

* in DUMP of SMD, DUMP0 is here at *. | ≠ LHLD WADR in DUMP

WAVE, etc on back page

```

4
4884 CDF247 CALL PUT - outputs a record
4887 ZA740C LHLD WRCO
488A 23 INX H
488B 22740C SHLD WRCO
488E 3E E6 MVI A, 0E6H - sync char
4890 CD 2C0C }
4893 CD 2C0C } PUSH OUT LAST BYTES
4896 CD 2C0C }
4899 AF XRA A }
489A D3 01 OUT 1 } TURN OFF MOTOR
489C C9 RETURN
  
```

* — Data ptrs and temp storage onward — *

```

WH3 EQU 0C2CH - tape I/O
WRCO EQU 0C74H - record number
WNAME EQU 0E6CH - file name
WLEN EQU 0C76H - file length
WADR EQU 0C77H -
??? EQU 0C7AH - bias addr
  
```

WNAME	6C	3180
	6D	
	6E	
	6F	
	70	
	71	
	72	
	73	-3187
WRCO	74	-3188
	75	-3189
WLEN	76	-3190
WADR	77	-3191
	78	-3192
WTYPE	79	-3193
??	7A	-3194
	7B	-3195

TEST#	-0-6 recs	FILENAME	TEST	0-2 intervals	0-5	0-3	0-4 - less than
		RCRD#	0	0-2	0-6	0-3	04
		LEN	0	0	1	0	212
		ADDR	0	20	40	30	40
		TYPE	2 (00)	5	2	5	5
		BIAS	73-177	73-177	73-177	73-177	73-177

0C05 C9 RET SP=FF8/HL=C04/DE=m/BC=DD7F/AF=247

30D5 CD A2 3B CALL 3BA2

3BA2 11 3A 49 LXI D, 493A

3BA5 D5 PUSH D - 493A

3BA6 CD AC 3C CALL 3CAC

3CAC AF XRA A ? expands LN is into ASCII?

3CAD 01 F0 D8 LXI B, D8F0 (HL) into decimal or ASCII

3CB0 CD D1 3C CALL 3CD1

3CD1 D5 PUSH D - 493A

3CD2 16 FF MVI D, FF

3CD4 22 A4 48 SHLD 48A4 put C04 at D03² 48A4

3CD7 14 INR D

3CD8 09 DAD B HL + BC = HL

3CD9 DA 04 3C JC 3CD4 C04 + D8F0 = E4F4

3CDC 2A A4 48 LHLD 48A4 → D8F0 jump

3CDF 42 MOV B, D → HL = C04

3CE0 01 POP D DE = 493A

3CE1 B0 ORA B

3CE2 C8 RZ returned

3CB3 01 18 FC LXI B, FC18

3CB6 CD D1 3C CALL 3CD1

3CE3 -not zero 3E 30 MVI A, '0'

3CE5 80 ADD B A = 33

3CE6 12 STAX D DE = 493A

3CE7 13 INX D

3CE8 C9 RET

3CB9 01 9C FF LXI B, FF9C

3CBC CD D1 3C CALL 3CD1 - put a '0' into 493B

3CBF 01 9C FF PUT a '7' into 493C

3CC2 CD D1 3C LXI B, FFFC

3CC5 01 9C FF PUT a '6' into 493D

3CC8 CD D1 3C RNZ not zero

3CCB C0 MVI A, CR

3BA9 3E 0D STAX D / DE = 493E - put a CR at end of 3070

3BAB 12 01 POP D

3BAC 21 BE 48 LXI H, 48BE

3BAD 3B BB 0E PUSH H

3BB1 CD 59 43 CALL 4359

3BB4 on page 4

page 1

HL = # to be converted
DE = where to put
at END, DE points to 1 byte after #

HL = C04₁₆ = 3076

- put a '3' into 493A

4359 AF

XRA A

page 2

435A 32 76 49

STA 4976

435D 32 75 49

STA 4975

4360 32 77 49

STA 4977

4363 3E F7

MVI A, F7

4365 32 79 49

STA 4979

4368 1A

LDAX D

4369 FE 2B

CPI '+'

436B CA 8243

JZ 4382

436E FE E3

CPI + token

positive number routine

4370 CA 82 43

JZ 4382

4373 FE E5

CPI - token

4375 CA 70 43

JZ 4370

4378 FE 2D

CPI '-'

437A C2 83 43 JNZ 4383

437D negative routine ~~~~~

4383 positive routine

4383 - 22 7A 49 SHLD 497A HL=48BE

4386 3E 04

MVI A, 4

4388 2B

DCX H

HL = 48BD

4389 36 00

MVI M, 0

438B 3D

DCR A

A=3

438C 12 88 43

JNZ 43 88

438F 22 7C 49

SHLD 497C

HL=48BA

4392 EB

XCHG

HL=493A / DE=48BA

4393 22 7E 49

SHLD 497E

4396 CD 55 44

CALL 4455

4455 2A 7E 49

LHLD 497E

HL now = 493A!

4458 7E

MOV A, M

A=33

4459 32 78 49

STA 4978

445C 23

INX H

HL=493B

445D 22 7E 49

SHLD 497E

4460 FE 3A

CPI ':'

4462 00

RNC

4463 FE 30

CPI '0'

4465 3F

CMC

4466 00 RNC

4467 EB 0F ANI 0F 33-03

4469 37

STC

446A C9

RET

check if A=11 0-9
if so, convert to binary
w/ binary set

4399 DA C2 43 JC 43C2

?error rtn.?

DEC PT HANDLER = 4402

EXPONENT " = 4416

43C2 CA BF 43 JZ 43BF

3C5 DA D0 43 JC 43 D0

Page 3

43D0 C0 6B 44 CALL 446B

43D3 21 77 49 LXI H, 4977

43D6 34 INR M

43D7 C0 55 44 CALL 4455

43DA DA D0 43 JC 43D0

43DD FE 2E CPI '0' decimal point

43DE CA 02 44 JZ 4402

43E2 FE 45 CPI 'E' exponent

see previous page

43E4 CA 16 44 JZ 4416

43E7 AF XRA A

43E8 21 77 49 LXI H, 4977

43EB 86 ADD M A=04

43EC C6 40 ADI 40

43EE FA A7 43 JM 43A7

43F1 21 76 49 LXI H, 4976

43F4 B6 ORA M

43F5 2A 7A 49 LHL D 497A HL=480E

A=F8

HL and DE now = 48BA

446B 47 MOV B, A

446C 21 79 49 LXI H, 4979

446F 34 INR M

4470 7E MOV A, M

4471 2A 7C 49 LHL D 497C

4474 CA 8B 44 JZ 448B

4477 F0 RP

was minus

4478 0F RRC

A was F8, now 7C

4479 DA 83 44 JC 4483

was minus

447C 78 MOV A, B

A and B = 03/

447D 07 RLC

A=6

447E 07 RLC

A=C

447F 07 RLC

A=18

4480 07 RLC

A=30 or 10* original A

4481 77 MOV M, A

HL=48BA

4482 C9 RET

43F8 77 MOV M, A

43F9 2A 7E 49 LHL D 497E HL=493F

43FC EB XCHG

HL=48BA

BC=06FF

43FD 3A 78 49 LDA 4978

DE=493F

AF=4406

4400 47 MOV B, A

A ← 0D

B ← 0D

4401 C9

to 3BB4 RET on first page

3BB4 E1

POP H

HL=48BE

PAGE 4

3BB5 C3 10 39 JMP 3910

DE=493F

BC=00FF

AF=0006

AFTER EXECUTION

3910 E5

PUSH H

SP

HL

DE

BC

AF

M

3911 11 FB FF

LXI D, FFFB

FF6

48BE

493F

00FF

0006

44

3914 CD 33 39

CALL 3933

FF4

FFFB

3917 01

POP D

7BFE

48BE

F983_{cm}

44

3918 EB

XCHG

48BE

7BFE

3919 CD FB 38

CALL 38FB on p. 7

391C From 3905 p. 7 AF

XRA A

FF6

48B9

7BF9

0000

3057_{cz}

4B

391D 3C

INR A

0046₂

391E 32 47 48

STA 48A7

0102

3921 C9 to 3008 on page 7 *

3925 From 3905 p. 7 2707 on pg 7

FFE

4B56

7BF9

2400

0102

00

3926 EB

XCHG

7BF9

4B56

3926 2A 5F 49

LHLD 495F

7BF9

3929 01 05 00

LXI B, 0005

7BFE

392C 09

DAD B

392D 22 5F 49

SHLD 495F

3930 C3 FB 38

JMP 38FB

adds

5 to DW 4955 and puts it back into 495F but saves DE

3933 2A 5F 49

LHLD 495F

FF2

48BE

FFFB

00FF

0006

44

3936 E5

PUSH H

7BFE

44

3937 19

DAD D

FF0

FF0

7BF9

FFFB

00FF

0006

44

3938 CD 64 39

CALL 3964

F983_{cm}

73

393B 22 5F 49

SHLD 495F

393E E1

POP H

FF2

7BFE

44

393F C9

RET to 3917

3964 05

PUSH D

FEE

FEC

7BF9

FFFB

00FF

0007_c

44

3965 EB

XCHG

FFFB

7BF9

3966 21 B8 48

LXI H, 48B8

48B8

57

3969 CD DB 39

CALL 39DB on next page

FEC

48B8

7BF9

00FF

7B16

57

396C DA 78 39

JC 3978

396F 21 5F 49

LXI H, 495F

495F

FE

3972 CD DB 39

CALL 39DB

F983_{cm}

3975 EB

XCHG

7BF9

495F

73

3976 01

POP D

SP HL DE BC AF M

FEE 7BF9 FFFB 0DFF F983

PAGE 5

cm 73

3977 08

RC to 393B on P4

3980 06 29

MVI B, 29

FF6 48B9 7BF9 0000 0102 4B

get ready
for compare

398F CD 9E 39 CALL 399E

3992 B8

CMP B

4B0E

2912 0D

3993 C8

RZ returned to 300B-P.7

2956 Z

3994 C3 D1 21

JMP 2101 - Syntax error routine

from 2D8A

3997 CD 9E 39

CALL 399E - advance
to next char

FF6 4B0E 7BF9 2900 2956 Z 0D

0D12

399A 2B

DCX H

399B C3 3538

JMP 3835 - on page 8

399E 2A B4 48

LHLD 48B4

FF4 4809 7BF9 2900 0102 4B

29

39A1 7E

MOV A, M

2902

39A2 23

INX H

4B0E

0D

39A3 FE 20

CPI 'B' } check for
spaces

2916

39A5 CA A1 39

JZ 39A1

2912

39A8 FE 09

CPI HT } check for
HT's

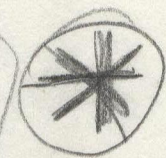
39AA CA A1 39

JZ 39A1

39AD C3 3538 - P8

JMP 3835

on page 8 - stores HL char ptr at 48B4 and RETS.

ADVANCE
TO NEXT
NON BLANK or
NON HT
CHAR

39B0

MVI A, ','

PUSH PSW

CALL 399E above

POP PSW

DCX H

CMP M

RZ

STC

JMP 3835

3DB9 - 0E 0D 20 CALL 20ED - put B into A and check for esc, then JMP WH1
+ more

39C4	0E 0D	MVI C, CR	39C9	0E 22	MVI C, 'm'
39C6	C3 CB 39	JMP 39CB			MOV A, m
39C9	0E 22	MVI C, 'm'			
39CB	7E	MOV A, m			
39CC	47	MOV B, A			
39CD	B9	CMP C			
39CE	C8	RZ			
39CF	FE 0D	CPI CR			
39D1	CA 01 21	JZ SYNTAX ERROR			
39D4	C0 B9 3D	CALL 30B9			
39D7	23	INX H			
39D8	C3 CB 39	JMP 39CB			
39DB	23	INX H			
39DC	7A	MOV A, D			
39DD	BE	CMP m			
39DE	2B	DCX H			
39DF	C0	RNZ			
39E0	7B	MOV A, E			
39E1	BE	CMP m			
39E2	C9	RET to 3975			
39E3	F5	PUSH PSW			
	7E	MOV A, m			
	23	INX H			
	66	MOV H, m			
	6F	MOV L, A			
	F1	POP A			
	C9	RET			

*
First call 20ED which
MOV A, B
CPI ESC
CZ 20AD - CAN rtn
JMP WH1

FEA 48B8 7BF9 0DFF 0007c 57
48B9 7B07c 4B
7B16
48B8 57
REC second time zero flag was set
495F
Compare DW pointed to by HL with DE
7B56 2 FE
F956 2
F983cm

38FB

FF4 48BE 7BFE 00FF F83 44

38FB 0E 05 mvi C, 05
38FD 7E mov A, m
FE 12 STAX D
38FF 2B DCX H
3900 1B DCX D
3901 0D DCR C

48BD 00
7BFD 0004

4483cm

3902 C2 F038 JNZ 38FD
3905 C9 RET to 391C

48B9 7BF9 0000 3057cz 4B

30D8 CD 8D 39 CALL 398D p.5
30DB C3 8A 2D JMP 2D8A

FF8 48B9 7BF9 0000 0102 4B

2D8A CD 97 39 CALL 3997

4BDE 7BF9 2900 2956 0D

2D8D FE E1 CPI token

FF8 4BDE 7BF9 2900 0D12 0D

2D8F D2 A0 2D JNC 2DA0

0D13C

2D92 FE C0 CPI unused token

0D17C

2D94 D2 ED 2D JNC 2DED

2D97 E1 POP H

FFA 2F01

01

2D98 3A A7 48 LDA 48A7

0117C

2D9B B7 ORA A set flags

0102

2D9C CA D1 2J JZ 2101 syntax error rtn

2D9F C9 RET to 2706

SP	HL	DE	BC	AF	M
FFC	2FD1	7BF9	2900	0102	01
FFE	4B56				

+2706 E1 POP H
2707 C3 25 39 JMP 3925

→ which returns to 26C0

26C0 CD 14 38 CALL 3814 - see pg 8
26C3 D2 BA 26 JNC 26BA

SP	HL	DE	BC	AF	M
1000	7BF9	4B51	0000	3056	73
1000	4B12	"	"	0916	20

26BA CD DB 3D CALL 3DDB - see p. 9
26BD CD C9 26 CALL 26C9 - below

00462

26C9 3E 0D mvi A, CR
26CB 32 A9 48 STA 48A9
26CE AF XRA A

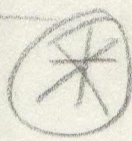
0D462
00462

26CF 32 A8 48 STA 48A8
26D2 CD 97 39 CALL 3997 p. 5
26D5 B7 ORA A set flags
26D6 F2 EF 26 JP 26EF advance to next char - 1 and store

4B13

9282m 92

continued pg 9



FROM 39AD on page 5 n 399B p.5
3835 22 B4 48 SHLD 48B4
3838 C9 RET to 3992-p.5

SP	HL	DE	BC	AF	M
FF4	4BDE	7BF9	2900	2912	0D

STORE CHAR PTR

3814 ^{nom pg 7 26C0} C0 7E 39 CALL 399E
3817 FE 5C CPI '\'
3819 C8 RZ
381A 7E 0D CPI CR
381C CA 2C 38 JZ 382C
381F

FFE 7BF9 4B51 0000 3056 73
advances HL to next non blank/HT
4BDF - now at 1st char of 2nd line
looking at char. before 4BDF
0097cm 0B
0D56 2 0B

382C 7E (GOOD MATCH CHAR) MOV A,M
382D 06 01 SUI 1
382F C6 FF ADI 255 ^{Pike -1 but sets carry flag}
3831 3F CMC
3832 02 42 25 JNC 2542
3835 at top of page

FFE 4BDF 4B51 0000 0D56 0B
0B56
0A16
0917C
0916
goes to 3835 if M=01 (end of program)

2542 23 23 23 INX H part 1st byte
2543 23 INX H part LN #1
2544 23 INX H part LN #2 to 1st program char
2545 22 B4 48 SHLD 48B4 ^{STORE CURRENT CHAR PTR}
2546 C9 RET to 26C3

14
00
20

~~0000~~

FROM 26BA →

3DDB 3A 90 48 LDA 489D

30DE B7 ORA A set flags

30DF C8 RZ to 26BD

SP	HL	DE	BC	AF	M	
FFE	4B12	4B51	Ø	Ø916	2Ø	
				ØØ16		
				ØØ46 ₂		

26D9 FE 97 FROM 26DB of p. 7
CPI not used token-mess

26DB D2 D1 Z1 JNC 21D1 - syntax error - carry set if under 97 - tokens at 80-97H are beginning of line tokens

26DE 11 51 2E LXI D, 2E51

26E1 CD 9E 39 CALL 399E - advance HL to next char loc

26E4 E6 1F ANI 1FH

26E6 Ø7 RLC

26E7 6F MOV L, A

26E8 26 ØØ MVI H, Ø

26EA 19 DAD D

26EB CD E3 39 CALL 39E3

39E3 F5 PUSH PSW

39E4 7E MOV A, M

39E5 23 INX H

39E6 66 MOV H, M

39E7 6F MOV L, A

39E8 F1 POP PSW

39E7 C9 RET

26EE E9 PCHL - jmp 2AF9

26EF - handles non-token chars.

2E3B thru 2E7E

2E51 to 2E7E for 2000

seems to be a DW

table for

2E14 to 2E35 or 2E36

has intermediate tokens

2E37-2E38 and maybe 2E39-2E3A seem available for DW use

92 becomes 1216

2416

4B14

4B24

ØØ24

2E75

2E76

2A76

2AF9

2416

9A

F9

F9

39

	A ← token
ANI 1F	A ← 1F AND TOKEN
RLC	A ← A * 2
MOV L, A MVI H, Ø	HL ← A
DAD D	HL ← A + 2E51 HL ← DW at HL
PCHL	PC ← HL

A > 00001111

AND '0001 1111' with A

1000 0000 = 80

0 0 0 0 = 0

0 0 0 0 = 0

can redo LET and it's DW for another new command

2E51 = LET DW

2E7D = DEF DW

ØØ 1001 0110 = 96

0001 1111

0001 0110 = 18

101110 2C

2E51

2E7D

Tables

POLY

MY EXTENSIONS, PROPOSALS

North Star

80 LET
 81 FOR
 82 PRINT
 83 NEXT
 84 IF
 85 READ
 86 INPUT
 87 DATA
 88 GOTO
 89 GOSUB
 8A RETURN
 8B DIM
 8C STOP
 8D PLOT
 8E RESTORE
 8F REM
 90 FN
 91 DEF
 92 !
 93 ON
 94 OUT
 95 POKE
 96 EXIT
 97 line # header
 98 VAL
 99 ASC
 9A CHR\$
 9B ELSE
 9C TAB
 9D THEN
 9E TO
 9F STEP
 A0 RUN
 A1 LIST
 A2 VERIFY
 A3 SCR

END

GOTO

WHILE

CHAIN

LE

TROD

TROD

TROD

LET
 FOR
 PRINT
 NEXT
 IF
 READ
 INPUT
 DATA
 GOTO
 GOSUB
 RETURN
 DIM
 STOP
 END
 RESTORE
 REM
 FN
 DEF
 !
 ON
 OUT
 FILL
 EXIT
 OPEN
 CLOSE
 WRITE
 CHAIN
 LINE
 RUN
 LIST
 NULL
 SCR

POLY

A4 CLE

A5 LOAD

A6 CON

A7 LINE (EDIT) EDIT

A8 REN

A9 FLIP

AA SAVE

AB STR\$

AC

AD

AE

AF

B0

B1

B2

B3

B4

B5

B6

B7

B8

B9

BA

BB

BC END

BD

BE

BF

C0

C1

C2

C3

C4 SQRT

C5

C6 INT

C7

XREF

AUTO

APPEND

FLIP

unused or BYE

ERROR

MUSIC

WHILE

CHAIN

FILE

TRACE

BLINK

VAL

TSTR\$

TOC

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

XREF LINE
XREF VAR1

FLIP

COMMANDS
STATEMENTS

North Star

CREATE

LOAD

CONT

REN

DUMP

SAVE

BYE

EDIT

DELETE

STEP

TO

THEN

TAB

ELSE

CHR\$

ASC

VAL

STR\$

NOEND MARK

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

STR\$

INT

POLY

North Star

C8

C9

CA SGN

CB SIN

CC LEN

CD CALL

CE RND

CF

D0

D1

D2

D3

D4

D5

D6

D7

D8 FREE

D9 INP

DA PEEK

DB ABS

DC COS

DD LOG

DE EXP

DF TIME

E0 (

E1 ^

E2 *

E3 +

E4 -

E5 -

E6

E7 /

E8

E9

EA

EB

LEN
CALL
RND

ATN

FREE
INP
EXAM
ABS
COS
LOG
EXP
TYP
(or [

*

+

-

-

/

EC	POLY		North Star
ED	AND		AND
EE	OR		OR
EF	>=		>=
F0	<=		<=
F1	<>		<>
F2	=>		
F3	=<		
F4	<		<
F5	=		=
F6	>		>
F7	NOT		
F8		OPEN	
F9		CLOSE	
FA		FOD	
FB		LINE	
FC		VARI	
FD		OFF ~ INSTR	
FE		unused	
FF	end of table		

} null tokens used like THEN, STEP

Make a new Token look up table at 2E5B on to 2FB8
put in new Cmds and file tokens

put new command vector DWs at 2E51 thru 2E5A

put old statement that can begin a line vector DWs at 49BD or after
change 26DF to 5

change 3642 to FE BD for STR#

Vector Table for Reserved Words that can start a line

LOC	Reserved WORD	VECTOR	Comment	Extensions
2E3B	RUN	2640	!CALL(9792, 0	no end parentheses
2E3D	LIST	25CF	!CALL(9679, 1st line number of program	
2E3F	VERIFY	45FA		
2E41	SCR	2518		
2E43	CLE	2520		
2E45	LOAD	45FB		
2E47	CON	2621		
2E49	LINE	03A9	- RET	EC, EDIT, VARI
2E4B	REN	2549		
2E4D	Flip	253B		
2E4F	SAVE	4768		
2E51	LET	26EF		ERROR or CHAIN or GET
2E53	FOR	27DA		
2E55	PRINT	2AF9	(99)	PUT/!
2E57	NEXT	2788		
2E59	IF	27F2		
2E5B	READ	2AA9		
2E5D	INPUT	2CDB		
2E5F	DATA	29D6		
2E61	GOTO	28AB		
2E63	GOSUB	2905		
2E65	RETURN	2936		
2E67	DIM	29F6		
2E69	STOP	2A5B		
2E6B	PLOT	30F5		
2E6D	RESTORE	2AEA		

LOC	Reserved WORD	VECTOR	Comment	Extensions
2E6F	REM	29F1		
2E71	FN	2CD4	FNEND <u>only</u> can start a line	
2E73	DEF	2CCB		
2E75	!	2AF9		
2E77	ON	28BE		
2E79	OUT	2CDD		
2E7B	FORE	2D21		
2E7D	EXIT	28EE or 28EF		

2E7F on is TOKEN table

Token Conversion Table

(11903₁₀ - 12216₁₀)
2E7F₁₆ - 2FB8₁₆

	Token Character			Token's Address		Reserved Word	Vector Table		
	Char	Dec	Hex	Dec	Hex		Addr (Hex)	Vector (Hex)	
CHAIN ERROR music etc	LET	α	128	80	11903	2E7F	LET	2E51	26EF
FOR	FOR	β	129	81	11907	2E83	FOR	2E53	270A
PRINT	PRINT	γ	130	82	11911	2E87	PRINT	2E55	2AF9
NEXT	NEXT	δ	131	83	11917	2E8D	NEXT	2E57	2788
IF	IF	ε	132	84	11922	2E92	IF	2E59	2752
READ	READ	ς	133	85	11925	2E95	READ	2E5B	2AA9
INPUT	INPUT	η	134	86	11930	2E9A	INPUT	2E5D	2CDB
DATA	DATA	θ	135	87	11936	2EA0	DATA	2E5F	2906
GOTO	GOTO	ι	136	88	11941	2EA5	GOTO	2E61	28AB
GOSUB	GOSUB	κ	137	89	11946	2EAA	GOSUB	2E63	2905
RETURN	RETURN	λ	138	8A	11952	2EB0	RETURN	2E65	2936
DIM	DIM	μ	139	8B	11959	2EB7	DIM	2E67	29F6
STOP	STOP	ν	140	8C	11963	2EBB	STOP	2E69	2A5B
PLOT	PLOT	ξ	141	8D	11968	2EC0	PLOT	2E6B	30F5
RESTORE	RESTORE	ο	142	8E	11973	2ECS	RESTORE	2E6D	2AEA
REM	REM	π	143	8F	11981	2ECD	REM	2E6F	29F1
FN	FN	ρ	144	90	11985	2ED1	FN	2E71	2CD4
DEF	DEF	σ	145	91	11988	2ED4	DEF	2E73	2CCB
END	END	τ	146	92	11996	2EDC	END	—	—
!	!	υ	147	93	11998	2EDE	!	2E75	2AF9
ON	ON	φ	148	94	12001	2EE1	ON	2E77	28BE
OUT	OUT	χ	149	95	12005	2EE5	OUT	2E79	2CDD
POKE	POKE	ψ	150	96	12010	2EEA	POKE	2E7B	2D21
EXIT	EXIT	≈	151	97	12015	2EEF	EXIT	2E7D	28EE
STEP	STEP	Σ	152	98	12020	2EF4	STEP	—	—
TO	TO		153	99	12025	2EF9	TO	—	—

	Token Character			Token's Address		Reserved Word	Vector Table	
	Char	DEC	HEX	DEC	HEX		Addr (Hex)	Vector (Hex)
THEN	÷	157	9D	12023	2EF7	THEN	—	—
TAB	↑	156	9C	12028	2EFC	TAB	—	—
ELSE	←	155	9B	12032	2F00	ELSE	—	—
CHR\$	→	154	9A	12037	2F05	CHR\$	(363D) —	{ CPI 09AH JZ 368A
ASC	√	153	99	12042	2F0A	ASC	(2078) —	{ CPI 099H JZ 3189
VAL	Ω	152	98	12046	2F0E	VAL	(207D) —	{ CPI 098H JZ 3246
STR\$	+	171	AB	12050	2F12	STR\$	(3642) —	{ CPI 0ABH JZ 36A2
RUN	⋈	160	A0	12055	2F17	RUN	2E3B	2640
LIST	!	161	A1	12059	2F1B	LIST	2E3D	25CF
VERIFY	"	162	A2	12064	2F20	VERIFY	2E3F	45FA
SCR	#	163	A3	12071	2F27	SCR	2E41	2518
CLE	\$	164	A4	12075	2F2B	CLE	2E43	2520
LOAD	%	165	A5	12079	2F2F	LOAD	2E45	45FB
CON	&	166	A6	12084	2F34	CON	2E47	2621
LINE	'	167	A7	12088	2F38	LINE ^{EDIT}	2E49	03A9 (RET)
REN	(	168	A8	12093	2F3D	REN	2E4B	2549
Flip	)	169	A9	12097	2F41	Flip	2E4D	253B
SAVE	*	170	AA	12102	2F46	SAVE	2E4F	4768
(	\	224	E0	12107	2F4B	(	2FB9	(0F) 03A9
*	b	226	E2	12109	2F4D	*	2FBF	(0A) 3DEA
+	c	227	E3	12111	2F4F	+	2FC2	(06) 3F0C
-	e	229	E5	12113	2F51	-	2FC8	(06) 3EF5
/	g	231	E7	12115	2F53	/	2FCE	(0A) 401A
AND	l	236	EC	12117	2F55	AND	2FDD	(04) 3087
OR	m	237	ED	12121	2F59	OR	2FE0	(02) 3093
>=	o	239	EF	12124	2F5C	>=	2FE6	(04) 303C

	Token Character			Token's Address		Reserved Word	Vector Table	
	Char	Dec	Hex	Dec	Hex		Addr (Hex)	Vector (Hex)
<=	p	240	F0	12127	2F5F	<=	2FE9	(04) 3035
<>	q	241	F1	12130	2F62	<>	2FEC	(04) 302F
=>	r	242	F2	12133	2F65	=>	2FEF	(04) 303C
=<	s	243	F3	12136	2F68	=<	2FF2	(04) 3035
<	t	244	F4	12139	2F6B	<	2FF5	(04) 3019
=	u	245	F5	12141	2F6D	=	2FF8	(04) 3029
>	v	246	F6	12143	2F6F	>	2FFB	(04) 301F
NOT	w	247	F7	12145	2F71	NOT	2FFE	(0D) 309E
^	a	225	E1	12149	2F75	^	2FBC	(0C) 3549
INT	F	198	C6	12151	2F77	INT	2FCB	(0F) 319D
LEN	L	204	CC	12155	2F7B	LEN	(2D6E)	{CPI 0CCH JZ 317C
CALL	M	205	CD	12159	2F7F	CALL	(2D73)	{CPI 0CCH JZ 30C0
RND	N	206	CE	12164	2F84	RND	2FE3	(0F) 35B2
SGN	J	202	CA	12168	2F88	SGN	2FD7	(0F) 30B4
SIN	K	203	CB	12172	2F8C	SIN	2FDA	(0F) 327B
SQRT	O	196	C4	12176	2F90	SQRT	2FC5	(0F) 3409
FREE	X	216	D8	12181	2F95	FREE	3001	(0F) 322E
INP	Y	217	D9	12186	2F9A	INP	3004	(0F) 31DD
PEEK	Z	218	DA	12190	2F9E	PEEK	3007	(0F) 323E
ABS	[	219	DB	12195	2FA3	ABS	300A	(0F) 30AF
COS	\	220	DC	12199	2FA7	COS	300D	(0F) 3271
LOG	]	221	DD	12203	2FAB	LOG	3010	(0F) 3443
EXP	^	222	DE	12207	2FAF	EXP	3013	(0F) 33B2
TIME	-	223	DF	12211	2FB3	TIME	3016	(0F) 30DE

Vector table for DIADIC operators and Numeric MONADIC (except for 'C')

OF leading byte means monadic?

2FB8 onwards	is	Token Table	Symbol	Token
2FB9	0F	J03A9	(ret)	E0
2FBC	0C	J3549	↑	E1
2FBF	0A	J30EA	*	E2
2FC2	06	J3F0C	+	E3
2FC5	0F	J34D9	SQRT	E4
2FC8	06	J3EF5	(minu)	E5
2FCB	0F	J319D	INT	E6
2FCE	0A	J401A	/	E7
2FD1	01	J0000	(4.0 mon)	(X8)
2FD4	0D	J30A8	(12456, 10) ?negate?	(X9)
2FD7	0F	J30B4	SGN	CA
2FDA	0F	J327B	(also used by COS)	CB
2FDD	04	J3087	LED ?	EC
2FE0	02	J3093	CALL ?	ED
2FE3	0F	J35B2	RND	CE
E6	04	J303C	>=	EF
2FE9	04	J3035	<=	F0
EC	04	J302F	<>	F1
EF	04	J303C	=>	F2
F2	04	J3035	=<	F3
F5	04	J3019	<	F4
F8	04	J3029	=	F5
FB	04	J301F	>	F6
2FFE	0D	J309E	NOT	F7
3001	0F	J322E	FREE	D8
4	0F	J31DD	INP	D9
7	0F	J323E	PEEK	DA

complect
MESSIT

```

1A LDAX D
E7 OR A A
CB RZ
EE 80 XRI 80
12 STAX D
C9 RET
  
```

LDAX D
ORA A
+ mrc

LDAX D
ORA mrc

CALL 3044
RNC
mov m, A
RET
CALL 3044
RZ
RC / mov m, A / RET
CALL 3044
RZ
mov m, A
RET

CALL 3044
RC
mov m, A
RET

CALL 3044
RZ
mov m, A
RET

CALL 3044
mov m, A
mvi A, A
RZ
mov m, A
RET

addr	?	DW vector	reserved word or symbol	token
300A	0F	J30AF	ABS	DB
0D	0F	J3271	COS	DC
10	0F	J3443	LOG	DD
13	0F	J33B2	EXP	DE
3016	0F	J30DE	TIME	DF

1A LOAD
 E67F ANI 7FH
 12 STACK
 C9 RET

3016 TIME
 TIME FN

3019 on 0 code 3019 = '<' code

3044 is the relational tester
 Z set if equal
 C set if <

strange Alphabets at 337B to 33A0

04 leading byte

means relational i.e. =, <=, etc
except AND isn't a relational

0F " "

means monadic operator except for 'C'
'C' might be monadic since what is inside
of parentheses reduces to 1 value

Unknown tables or constants

337B	15	N/AK	
337C	70	'P'	
337D	79	'Y'	
337E	63	'C'	} or J4163?
337F	41	'A'	
3380	50	'P'	
3381	0, 0, 0		
3384	40	'@'	} or J3040?
3385	30	'0'	
3386	0, 0, 0		
3389	42	'B'	
338A	J4243		
338C	J4894		
338E	J3340		
3390	J3012		
3392	J4685		
3394	10		
3395	82		
3396	J3670		
3398	C5		
3399	J2415		
	20		
	J4378		
	72		
	J2444		
	51, C0, 19		

OVER

76/73 / 78/69

Unknown tables or constants

33A4 J3887

6 51

7 3447

9 40

A J4270

33AE J2645 -

33AE J4289

33B0 J4313

33B2 on is code for EXP rtn

34D4 to 34D8 = a BCD Floating point format '1'

3176 to 317B inclusive is 20 to 8 4 2 101
table for PLOT?

PLOT goes from 30F5 to 3175?

OVER

Messages

STARTUP MESSAGE

Input error - retype"

202A



200A-2040 : "Poly 88 BASIC version 1A00.", FREE(0), "0

in 1 bytes free." CR

Stop"

Interrupted"

let men



: "Ext Poly 88 BASIC 1A00.", 0(0), "0 bytes free." CR

ERROR MESSAGES

200A - 2040 "Poly BASIC version 4.00", FREE (0), "bytes free." "C"

22D0 - 22D9 * Subscript "

22DA - 22E6 Bad Argument "

22E7 - 22F0 Dimension "f"

22F1 - 22FD function & Def "

22FE - 230B Out of bounds "

230C - 2310 Type "

2311 - 2317 format "

2318 - 2323 Line Number "

2324 - 2332 Divide by zero "

2333 - 2339 Syntax "

233A - 2348 Can't continue "

2349 - 2352 Double def "

2353 - 2361 Control stack "

2362 - 2366 Read " should be READ "

2367 - 2371 Complexity "

2372 - 2377 Input "

2378 - 2383 memory full "

2384 - 2390 Arg mismatch "

2391 - 239F Illegal direct "

23A0 - 23A6 Length "

23A7 - 23AF Overflow "

23B0 - 23C4 RETURN without GOSUB "

23C5 - 23D1 Missing NEXT "

23D2 - 23DA FOR - NEXT "

ERROR MESSAGES

23DB-23E3

Checksum"

23E4-23EA

Verify"

Input error - retype"

2E14

Error"

2E1B

in line"

Stop"

Interrupted"

Data
Pointers
and Temp.
Storage

495F

4890 - 48EA

storage

CZ = JNZ

CA = JZ

ZZ = SHLD

ZA = LHLD

4890 DB STA A=0 2175 LDA 2293 LDA 223F STA A=0 23F9 STA 4890

489E DW SHLD 1000/SP 2167 SHLD 1000/SP 2592 LHLD 20B3

48A0 DW INPUT BUFFER READOUT PTR

LDA 48A0 2080 SHLD HL=0CBF 20A8 LHLD 20BC SHLD HL=CBF 20D2

48A2 DW INPUT BUFFER WRITEN PTR

LHLD 207F SHLD 2094 SHLD HL=CBF 20AB LDA 48A2 20BF

48A4 DW SHLD 3CD4 LHLD 3C0C

48A6 DW 48A2 - if zero, then error

48A8 DB LDA 2225

48A9 DB CPM 21CC if = 0 then syntax error

48AA DW LHLD 2529 SHLD 213F LHLD 2213 SHLD 251D LHLD 250C

current top of text ptr

48AC DB STA A=1 2161 LDA 2249 LDA 223C if 0, then print error msg with a response to arrow

48AD DB LDA 2183 STA A=0 2537 - if = 0, we can't continue if < 0, we can continue

48AE DW LHLD 2CA4 SHLD 2C10 3 holds stack

48AF DW INPUT routine

48B0 DW SHLD 1000/SP 212A LHLD 2158 - Holds SP

48B2 DW SHLD HL=03A9 2123 LHLD 2007 - PIP or unflip input handling

48B4 DW SHLD 21B3 SHLD 2217 SHLD HL=48EA 23EF LHLD 23F9 SHLD 246E LHLD 247D LDA 48B4 24BA LDA 48B5 24BE SHLD 2545 LHLD 25A5 SHLD 4638

48B6 BASIC CURRENT PROGRAM PTR SHLD 4385

48B8 DW LHLD 4969 252D SHLD 222C - of variables to array + strings?

48BA ?

B ?

C ?

D ?

E ?

48BF DB MOV M,C 249E LINE length

48C0 DW SHLD 23F5 line #

48C2 DB 48C2 on Compressed input buffer from 48C2 to over

48B4 is DW pointing to current char ptr in BASIC program during execution; used for other purposes

:

48EA ~~DE~~ ^{40A} Uncompressed Input Buffer

480A to compressed input buffer

48EA to 4939 uncompressed input buffer (80 bytes)

34 unknown bytes from 493C to 4950
unknown from 496B to 49A0

EA }
CA }
DA }
1 }
2A }
3A }

$5 \times 16 = 80 \text{ bytes}$

CZ }
DZ }
EZ }

EA

CZ = JNZ
CA = JZ

ZZ = SHLD
ZA = LHLD

493A - 49AF storage

493A DW SHLD | LXD
25EC 3BA2

495F DW SHLD
2526

bottom of scratch space ptr
grows downwards

(scratch space ?
top of mem - ? on down)

4961 DW ?

holds current length of a string

4963 DW ?

holds current length of a string

4965 DW ?

holds max len of a string

4967 DW LHLD
2232

4969 DW LHLD
222C

SHLD 4973 at 4404

LXI H, 4977 at 4496 / LXI H, 4990 at 3F46

49A1 DW SHLD
216E | SHLD
254C | SHLD
2558 | LHLD
2585

49A3 DW SHLD
2171 | SHLD
254F | SHLD
256C | LHLD
2589

49A5 DW LHLD
25A1

49A6 ?

49A7 DB STA
45D
214C

49A8 DB STA
218D

49A9 DB STA
2111

49AA DB ?

(H)
COMPARED
TO SCREEN
TOP MEM
SOMETHING
W/ SCREEN
+ TOP MEM

? TIMER DW? — used by load

49AB DB STA
2246 | LDA
2267 | STA
24EC | LDA
24E4 | LDA
24C5 | STA
24D2

START OF TEXT
TOP OF BASIC

49AC DW LHLD
22BB | SHLD
2119 | LHLD
THEN SPHL
2130 | LHLD
2518 | LHLD
257E | LHLD
2589

49AE DW SHLD
211D | LHLD
2520 — TOP OF MEM

49B0 free RAM onward

3181
3200 - unflipped

CPU is $\frac{1}{2}$ K RAM used by BASIC

0C6C	DS	8	WNAME	File name
0C74	DS	2	WRCD	record #?
0C76	DS	1	WLEN	record length
0C77	DS	2	WADR	address? or ?
0C79	DS	1	WTYPE	file type? 5?
0C7A	DS	2	?	?
0C7C	DS	2	?	?
0C7E	DS	2	?	?

0C80-0CBF DS-64?

Circular Input Buffer for KSR at 2069H. Starts at 0CBF and goes downward.

0C54 is used by INP and OUT routines
to

0C56

INP port \ RET

or

OUT port \ RET

Misc.

3006 prints a CR, then puts a $\emptyset$ at 48A6
399E Advances HL (LHLD 48B4) to next (non blank or non-HT char) + 1
A contains the character. Then jumps to 3835

3835 stores HL at 48B4 and RET
398D Compares next pgm char w/ ')' for match

3925 junk
3814-381C advances to next (non blank or non-HT) char, then checks for \ or CR

382C checks A, possibly something to do with Linett
2542 Incr HL past (1st byte + LN #) of a line, stores it at 48B4, RET

3997 Advances HL (LHLD 48B4) to next (non-blank or non-HT char), then A=char then 3835

26EF handles non token chars.

39E3 Puts DW pointed to by HL into HL, then RET (LHLD's (HL))
2AF9 Print routine Saves PSW

2609 converts a token to its routine's address

3BF5 multiplies HL by 10, then ands 0FH onto L

3D44 Relational tester

3ADB Compares DW pointed to by HL with DE

36FD puts DW pointed to by HL into DE

20B7 BASIC's WHO

2069 BASIC's KISR

3A92 HL = HL - DE

39C9 prints any string up to a " (error if embedded CR's)?

HL points to first character

38F8 moves a FP BCD '1' to (DE-4) thru (DE)

3CAC converts HL to decimal ASCII; DE points to where to put ASCII; at END, DE₁#+1
ASCII

PABM 3BB8 puts next expression into DE and A
an expression is like x, y, or z in a plot

39B0 checks that next character is a comma

39B2 checks that next charactn = Accum

4551 add A to HL

3724 IN - 48B4
A1 = 6
OUT - 48B4
points HL to a variable's 1st char. or ^{if string} ^{or point if number} _{DE = length of string}. ZERO flag set if a string variable
also used by functions and operators like VAL, ASC, *, +, etc (LET and ARRAYS)
(# of bytes per ch) - 1 for numbers
to be used; i.e. A#(2,3) DE = 0002
A# = "123456789" DE = 000A if old
A# is LEN = 000A

POINTERS TO EXPAND/COMPRESS TOKEN TABLE

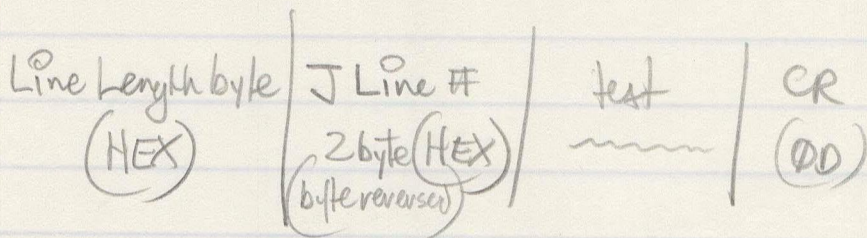
24D6H compress routine

24F6H expand routine

4658H to expand a file name if converted to a token

to make my own token table, just change these pointers and make sure there is a label for every token and vice versa

Program Line Format



A 97H precedes all line numbers after GOTO's, GOSUBS, IF...THEN(goto) Ln, etc

and the line numbers that follow is a 2 byte reversed binary. bytes in byte-reversed format

ADD BASIC

String and Number Storage Formats

STRING

Header byte	? DW	MAX LEN DW	CUR LEN DW	ACTUAL STR Chars
8FH if varname without following optional digit else 2nd nybble equals optional digit	? free? ptr?	Low Hi	Low Hi	

NUMERIC

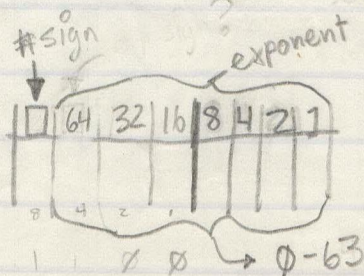
- 8 digit precision

(BCD)

HDR BYTE	? DW	MS Nybble	LS Nybble	Exponent/sign
same as for string only 0 instead of 8 at MS Nybble	? free? ptr?	4 bytes BCD		MSB is sign
				1 = 10/00/00/00/41 power -1 = 10/00/00/00/C1

after end of text (01) byte 0 at DW pointing to each string's hdr byte

? subtract 42H from 'exponent byte' to get exponent



$$-1 = -1E0 = 10/00/00/00/C1$$

$$1 = 1E0 = 10/00/00/00/41$$

$$.1 = 1E-1 = 10/00/00/00/40$$

$$.01 = 1E-2 = 10/00/00/00/3F$$

subtract 65 (41H) from bits 0-6 of the exponent byte to get real exponent

Var
Names
and other
extentions

Extensions Proposed or Completed

Commands

HELP	syntax assistance
TRACE ON	debugging aid - prints line number as executed
TRACE OFF	turns off TRACE
APPEND	loads a file after the current file, REN's accordingly
VARIABLES	lists VarNames
EDIT	edit a line
AUTO	automatic line numbering
FLIP	switch between 2 resident programs
XREF	

Statements

CHAIN	loads in a program
POS	returns the position of a string within a string
GET	data file get.
PUT	data file put.
MUSIC	PAIA music output
ERROR	on error goto capability
POS	find a string in another string (INSTR)

on error goto capability

ERRFLG = 1, add location
 2, add + line
 3, at 2225 put - JMP ERROR
 4, at 2225 put - JMP ERROR
 5, at 2225 put - JMP ERROR
 6, at 2225 put - JMP ERROR
 7, at 2225 put - JMP ERROR
 8, at 2225 put - JMP ERROR
 9, at 2225 put - JMP ERROR
 10, at 2225 put - JMP ERROR
 11, at 2225 put - JMP ERROR
 12, at 2225 put - JMP ERROR
 13, at 2225 put - JMP ERROR
 14, at 2225 put - JMP ERROR
 15, at 2225 put - JMP ERROR
 16, at 2225 put - JMP ERROR
 17, at 2225 put - JMP ERROR
 18, at 2225 put - JMP ERROR
 19, at 2225 put - JMP ERROR
 20, at 2225 put - JMP ERROR
 21, at 2225 put - JMP ERROR
 22, at 2225 put - JMP ERROR
 23, at 2225 put - JMP ERROR
 24, at 2225 put - JMP ERROR
 25, at 2225 put - JMP ERROR
 26, at 2225 put - JMP ERROR
 27, at 2225 put - JMP ERROR
 28, at 2225 put - JMP ERROR
 29, at 2225 put - JMP ERROR
 30, at 2225 put - JMP ERROR
 31, at 2225 put - JMP ERROR
 32, at 2225 put - JMP ERROR
 33, at 2225 put - JMP ERROR
 34, at 2225 put - JMP ERROR
 35, at 2225 put - JMP ERROR
 36, at 2225 put - JMP ERROR
 37, at 2225 put - JMP ERROR
 38, at 2225 put - JMP ERROR
 39, at 2225 put - JMP ERROR
 40, at 2225 put - JMP ERROR
 41, at 2225 put - JMP ERROR
 42, at 2225 put - JMP ERROR
 43, at 2225 put - JMP ERROR
 44, at 2225 put - JMP ERROR
 45, at 2225 put - JMP ERROR
 46, at 2225 put - JMP ERROR
 47, at 2225 put - JMP ERROR
 48, at 2225 put - JMP ERROR
 49, at 2225 put - JMP ERROR
 50, at 2225 put - JMP ERROR
 51, at 2225 put - JMP ERROR
 52, at 2225 put - JMP ERROR
 53, at 2225 put - JMP ERROR
 54, at 2225 put - JMP ERROR
 55, at 2225 put - JMP ERROR
 56, at 2225 put - JMP ERROR
 57, at 2225 put - JMP ERROR
 58, at 2225 put - JMP ERROR
 59, at 2225 put - JMP ERROR
 60, at 2225 put - JMP ERROR
 61, at 2225 put - JMP ERROR
 62, at 2225 put - JMP ERROR
 63, at 2225 put - JMP ERROR
 64, at 2225 put - JMP ERROR
 65, at 2225 put - JMP ERROR
 66, at 2225 put - JMP ERROR
 67, at 2225 put - JMP ERROR
 68, at 2225 put - JMP ERROR
 69, at 2225 put - JMP ERROR
 70, at 2225 put - JMP ERROR
 71, at 2225 put - JMP ERROR
 72, at 2225 put - JMP ERROR
 73, at 2225 put - JMP ERROR
 74, at 2225 put - JMP ERROR
 75, at 2225 put - JMP ERROR
 76, at 2225 put - JMP ERROR
 77, at 2225 put - JMP ERROR
 78, at 2225 put - JMP ERROR
 79, at 2225 put - JMP ERROR
 80, at 2225 put - JMP ERROR
 81, at 2225 put - JMP ERROR
 82, at 2225 put - JMP ERROR
 83, at 2225 put - JMP ERROR
 84, at 2225 put - JMP ERROR
 85, at 2225 put - JMP ERROR
 86, at 2225 put - JMP ERROR
 87, at 2225 put - JMP ERROR
 88, at 2225 put - JMP ERROR
 89, at 2225 put - JMP ERROR
 90, at 2225 put - JMP ERROR
 91, at 2225 put - JMP ERROR
 92, at 2225 put - JMP ERROR
 93, at 2225 put - JMP ERROR
 94, at 2225 put - JMP ERROR
 95, at 2225 put - JMP ERROR
 96, at 2225 put - JMP ERROR
 97, at 2225 put - JMP ERROR
 98, at 2225 put - JMP ERROR
 99, at 2225 put - JMP ERROR
 100, at 2225 put - JMP ERROR

IDEA - ~~not~~ implement additional Reserved Words via a TRAP in the SYNTAX ERROR routine

Addons to ADO Basic

ELIN(^{dummy var}~~n~~) returns line number of last error
ERM#(~~n~~) returns text of error message 'n'

HELP ~~reserved word~~ - displays a documentation file about 'reserved words'

PRINT @(^{LINE #}~~n~~, ^{COL #}~~m~~)
or TAB(^{LINE #}~~n~~, ^{COL #}~~m~~)

DEL - delete a line # ~~1~~ or line #1 thru line #2

Limit # of chars in an input; i.e. @(-10, n) type of thing

INPUT (MAX = 10) ["prompt"], var.....

FIND - searches for a string in a program, and then enters edit mode in line where string was found

BLOCK READ/WRITE

BYTE

CHAIN & SWAP

DELAY or PAUSE

LOAD, SAVE, REPLACE, DELETE, RENAME

also for SAVE & REPLACE, need a way to write out a comment record

i.e.: SAVE TEST[,P] "This is a comment"

APPEND/MERGE

LABELS

Label: statement -or- "label": statement

ON IKEY / ON ESC

ON ERR

RANDOMIZE

TINPUT

TRUN#, FILL#, SREP#, RPAD#, LPAD#

remove trailing characters from a string
fill a string with lots of the same characters
search for & replace a substring
right & left pad a string w/ blanks

~~SPEC~~
51.5K
84081.5
618500
4410.0K

785000.5

ADD a GOTO/GOSUB optimizer to

BASIC:

add a PRINT AT line #, column #:
function

when the optimization is in effect, the line number following the 97H (or whatever) for GOTOs and GOSUBs the 97H (or whatever) which ~~precedes~~ precedes will be replaced with the address of the beginning of the GOTO or GOSUB line number line instead of the line #. Also, the GOTO and GOSUB (and possibly THEN) routine addresses in the vector table would need to be changed to new addresses

Purpose - searches in a line for a line # matching BC. If it finds one, it prints the line # in DE, then continues searching until CR.

IN { HL - start of text ptr for line to be searched
DE - line # of line being searched
BC - line # to be found
A -

OUT { HL - EOL+1
DE - same
BC - same
A - changed

LINE	MOV	A, m	get a character
	INX	H	Incr ptr to next char
	CPI	CR	are we at EOL?
	RZ		return if so
	CPI	97H	found a line #?
	JNZ	LINE	try again if not
	PUSH	D	else save line #
	MOV	E, m	get 1st byte into E
	INX	H	then get next
	MOV	D, m	byte into D
	INX	H	for compatibility with next scan
	CALL	CMPIR	(DE=BC?) is it the line # we want?
	POP	D	get back line #
	CZ	PRINTDE	print the line # if equal
	JMP	LINE	then go back for more

Purpose - Print an HT, update HT counter, print DE as ASCII decimal. if

IN {
HL -
DE - line # to be printed
BC -
A -

OUT {
HL - unchanged
DE -
BC - unchanged
A -

PRTDET-
PUSH B
PUSH H
CALL TABCNT
LDA HTABCNT
ANI 0FH
CPI 8
CZ TABCR
CALL CONVERT
POP H
POP B
RET

print atab line # 1 2 3 4 5 6 7 8
look at count
turn off all bits except 8th, because 8 tabs = 1 line
if 8 and only 8 was on
ADD BASIC routine to convert binary reg to ASCII decimal
print the chars

5100
before call 3C07
LXI H, # to convert
value to get
char
CPI B
CD 07 3C

PRINTDE
LDA PRTFLAG
ORA A
JNZ PRTDET
PUSH D
MOV D, B
MOV E, C
CALL ~~PRTDET~~ CONVERT
POP D
CALL 4.0TAB not HREF Tab routines
MVI A, BKSAC
CALL WHI
CALL WHI
MVI A, '-'
CALL WHI
MVI A, I
STA PRTFLAG
CALL PRTDE

5000H

HTABCNT = 50A9

LINECNT = AA

PRFLAG = AB

5082 converted

(0000) 10 GOTO 20
(0014) 20 GOSUB 30
(001E) 30 IF X=1 THEN 10 ELSE 20

CONVERT

~~XCHG
LXI D, 493A
~~CALL 3C01~~
CALL 3CAC
RET~~

~~XCHG
LXI D, 493A
B, D8FD
CALL 3C01
CALL~~

after every call 3C01, zero flag is set if $HL < DE$ (have no # to put

CONVERT

XCHG
LXI D, 493A
LXI B, D8FD -10,000
~~CALL 3C01~~ CALL QUICK
CJNZ WH1
LXI B, FC18 -1,000
CALL QUICK
LXI B, FF9C -100
CALL QUICK
LXI B, FFF6 -10
CALL QUICK
LXI B, FFFF -1
CALL QUICK
JMP

QUICK CALL 3C01
CJNZ WH1
RET

Purpose - searches for a matching line # in a line. If it finds one, it prints the line # that it is in, then continues searching.

IN { HL = start of the line to be searched's text
 DE = line # to be searched
 BC = line # to be found

OUT { HL = EOL+1
 BC = same

LINE	MOV	A, m		get a character
	INX	H		incr ptr to next char
	CPI	CR		are we at EOL?
	RZ			return if so
	CPI	97H		found a line #?
	JZ	LINE	PUSH D	if not else save line # of line we are searching
	MOV	E, m		else get low order byte into E
	INX	H		and get high order
	MOV	D, m		byte into D
	INX	H		for next scan
	CALL	CMPR	POP D	see if DE = BC get back line #
	CZ	PRTDE		print the line # if equal
	JMP	LINE		



INPUT AB: ESC TO Inbr, TAB TO Inbr, CR TO Inbr

IN BC = line # to be found
 HL = start of current line's text
 OUT BC = line # to be found
 HL = EOL + 1

~~throughout DE = start of line~~
~~HL =~~

LINE	EQU	\$	HL = line # to be found, BC = start of line's text
	MOV	A, M	get a char
	INX	H	incr ptr to next char
	CPI	CR	are we at EOL?
	RZ		return if so
	CPI	97H	found a line #?
	JZ	LINE	if not
	MOV	E, M	get low order byte
	INX	H	incr so we can
	MOV	D, M	get high order byte
	INX	H	to be compatible for next search
	CALL	CMR	see if DE = BC
	CZ	PRTDE	print DE in decimal if equal
	JMP	LINE	

IN	BC = line # to be found
OUT	BC = line # to be found
	zero flag set if nothing printed

SCAN	LHLD	?49ACH?	get HL = start of text
SCANIT	CALL	LINE	search thru a line
	MOV	A, M	get EOL+1 char
	CPI	01	see if at end of text
	JNZ	SCANIT	if not, scan some more
	LDA	PRTFLAG	else see if anything was
	ORA	A	printed for this line #
	RZ		if nothing was printed
	LDA	HTABCNT	get amt of flags
	SHIFT		divide by 8 to get # of lines
	RITE		
	FOUR		
	TIMES		
	PUSH	H	

LINE # XREF

```

1 XRA A
3 LXI --
1 MOV M,A
1 INX H
1 M
1 M

```

SETUP

XRA

A

zero A, so we can

LXI

H, HTABCNT

MOV

M, A

zero HTABCNT

INX

H

MOV

M, A

zero LINECNT

INX

H

MOV

M, A

zero PRTFLAG

LHLD

? 49ACH?

get start of text into HL

AGAIN

LDA

LINECNT

get line count into A

CPI

? 15 or 16?

are we at bottom of screen?

JNZ

AGAINOK

if not

XRA

A

zero A, so we can

STA

LINECNT

zero line count

CALL

WH0

wait for user input

CALL

CLEAR

for next screen full

AGAINOK

SHLD

LINSTR

finally store start of line addr

INX

H

incr over line length byte

MOV

D, H

put that #

MOV

E, L

DE as well as HL

CALL

LHLD(HL)

get DW pointed to by HL

XCHG

HL pts to line #, DE = line #

INX

H

increment past

INX

H

line number

LDA

PRTFLAG

get print flag

ORA

A

to see if we printed

JZ

NOCR

before

CALL

CROUT

otherwise print a CR,

XRA

A

then zero (initialize)

STA

HTABCNT

tab count and

STA

PRTFLAG

and print flag

NOCR

MOV

A, M

get a char

INX

H

but also bump pointer over

CPI

CR

was it a CR?

JNZ

LOOK

if no CR, not at line end yet

LDA

PRTFLAG

also get print flag

ORA

A

set flags

	JNZ	NOPRT
	PUSH	H
	LDA	HTABCNT
	SHIFT	
	RITE	
	FOUR	
	TIMES	
	LXI	H, LINECNT
	ADD	M
	MOV	M, A
	POP	H
NOPRT	MOV	A, M
	CPI	01
	JNZ	EOL
	JMP	CROUT
EOL	LHLD	LINSTR
	MOV	E, M
	MVI	D, 0
	DAD	D
	JMP	AGAIN
LOOK	CPI	97H
	JNZ	NOCR

to see if we printed anything
save H
get Tab count, divide by
8 to get # of lines printed
for this line #

add current line cnt to prev.
and put new # back

get char after CR
see if end of text
if not, else
print a CR + return to BASIC
get start of finished line
get line length into DE
(zero = 0)
get to next line addr.
then do next line

found a line #?

LINE	finds 97H + line # in a line for a specified line #
SCAN	finds " " " all lines " " " "
INCR	gets into HL the next line # to be found

PRTDE → prints an HTAB, incr HTABCNT, convert DE to
ASCII decimal, print it, RET
1st, if no line # was printed, prints it and sets PORTFLAG
LINE # 00-

↑
HERE GOES
TO HERE


```

TABCR    CALL CR not CROUT
TAB      CALL TAB4.0
          PUSH H HTABCNT
          LXI H, HTABCNT
          INR m
          POP H
          RET

```

print a tab
 save HL
 point to HTABCNT
 increment HTABCNT
 get back HL

```

TABCR    PUSH H
          LXI H, LINECNT
          INR m
          POP H
          JMP TAB

```

```

CR →    CALL CROUT
           LDA LINECNT
           INR A
           CPI 15
           JNZ CREND
CREND    STA LINECNT
          RET
HOLD      CALL WHD ←
           XRA A
CREND      STA LINECNT
           RET

```

```

CMR      MOV A, B
          CMP D
          RZ
          MOV A, C
          CMP E
          RET

```


AUTO line number

INITIAL DE 2
LINENBR DW 10
LININCR DW 10

AUTO ; after the initial space is detected

~~LDA INITIAL ORA #32 NOAUTO (if LINENBR & LININCR not initialized)~~

LHLD LINENBR \XCHG\ LHLD LININCR \PAD 0\ JC 'OVERFLOW ERROR'

CALL portion of expand which does line #s (put it at 48EA?)

MVI M, 0 assume HL = end of line # + 1

LXI H, 48EAH

AGAIN MOV A, M

ORA A

JZ to rest of normal input routine

CALL WH1 print out the AUTO line number

INX H

JMP AGAIN

~~OVERFLOW XRA A
STA INITIAL
JMP 'overflow error'~~

~~LDA~~

~~JC~~

~~SHLD~~

NOAUTO

need to have 2000 ins to initialize LINENBR + LININCR

AUTO CMD

input #s just like REN where 2nd # is optional

(CALL PARAM SHLD LINENBR)

if 2nd # is missing → LXI H, 10 decimal

SHLD LININCR store it as the increment

Syntax

AUTO starting-line-number, optional increment between lines

LINXREF
LINAGIN

LHLD 49AC?
PUSH H
INX H
MOV C,m
INX H
MOV B,m
CALL SCAN
POP H
MOV E,m
MVI D,0
DAD D
MOV A,m
CPI 01
JNZ LINAGIN
JMP CROUT

get start of test pointer

point to line #

get line # DW
into BC

put length byte into E
zero D

HL = start of next line

get the char
EOT ext.?

else print ack and use it's RET

PURPOSE Scan thru all of text for a line # matching BC. If it finds one, it prints the line # in DE, then continues searching until end of text.

IN { HL -
DE -
BC - line # to be found
A -

OUT { HL -
DE -
BC - same
A - ~~zero flag set if nothing printed~~

```
SCAN  
SCANIT  LHLD 49AC?  
        INX H  
        MOV E, M  
        INX H  
        MOV D, M  
        INX H  
        CALL LINE  
        MOV A, M  
        CPI 0J  
        JNZ SCANIT
```

if SOMETHING was printed, need to back CR

```
RET  
ORA A  
RZ  
LDA HTABCNT  
  
LXI H, LINECNT  
ADD M  
MOV M, A  
CPI 14  
RNC
```

```
LDA PRFLAG  
ORA A  
RZ  
XRA A  
STA HTABCNT  
STA PRFLAG  
JMP CR and use it's RET
```

get HL = start of text
incr past length byte to line # DW
get low order byte into E
point to high order byte
get it into D
to be compatible with LINE
find and print any matches in this line
get EOL+1 char
see if at End of Text
do some more if not at end
else RET

~~if print flag
to see if anything was printed
if nothing was
else print a CR
see how
many lines
were printed
so far
point to line counter
add it to HTABCNT
and put it back
have we printed 14 lines?~~

— reset HTABCNT, PRFLAG

Notes of I intend to sell my
super basic, change 'flip' to 'case'.

STEP ^{ON}/_{OFF} - like TRACE ^{ON}/_{OFF}, but single-steps a
line (or statement) at a time

DATE~~##~~ = "<mm/dd/yy>" - SBL to set DATE~~##~~

DATE~~##~~ (<dummy arg>) - function to return the date as a string

NEW - same as SCR

TOD~~##~~ = "<hh:mm:ss>" - SBL to set the time-of-day

TOD~~##~~ (<dummy arg>) - function to return the time-of-day

RANDOMIZE - same as ~~TIME~~ RND(TIME(1)/65535)
RND(65536/TIME(1))

UPPER~~##~~ (<string>) - returns <string> converted to upper case

ADDR(<variable-name>) - returns the address where <variable-name> is stored

LOAD(<address>) - returns dataword at <address>. (can be an expression)

make up a GOTO & GOSUB table for faster branches. Build
table right after typing RUN.

DEL <line #1> [, <line #2>] - delete <line #1> or delete
<line #1> thru <line #2>. (will need to reset top of text and then do a CE)

COPY <line #1> <line #2> ~~##~~, <line #3> - copy <line #1> thru <line #2>
right after <line #3>

RTC ^{ON}/_{OFF} - enable/disable RTC routines BLINK ^{ON}/_{OFF}, TOD, BLINK ^{ON}/_{OFF}
Disabling reduces the RTC overhead. When RTC flag = OFF,
any use of BLINK, TOD, ~~TIME~~, DATE~~##~~ will cause a
FUNCTION DISABLED ERROR. ^{maybe music too} RTC flag has been set ON.

RTC ON will cause the following to be printed. "TOD~~##~~ and DATE~~##~~ must be set."

RTC OFF
BLINK ON
BLINK OFF

TRACE ON
TRACE OFF
STEP ON
STEP OFF

AUTO ON
Auto OFF

"RTC flag ~~OFF~~" has been set OFF.
BLINK has been set ON
"
TRACE ON
"
OFF
STEP ON
"
OFF

in
immediate
mode

Notes

BYE - exits to monitor (maybe also allow BYE <filename>[$\frac{B}{P}$] which reads in <filename>.)

EDIT <line#>

EDIT <"search-string">[, <starting-line#>] - edits the first line with <search-string> in it, or first line starting at <starting line#>
↑ ~~must~~ must do a tokenized search and an untokenized search

POS () - also allow quoted variables make INSTR~~xx~~ and SUBSTR~~xx~~ in input be changed to POS

XREF ^{LINE} _{VARI}

make printed or input CTRL-G's output to PA/A

MUSIC - maybe use delay loops, not RTC

AUTO ~~xxx~~ - starts automatic line numbering

~~AUTO [starting line#] [increment]~~

AUTO [starting line#][, <increment>]]

END as the only input for an AUTO numbered line ends AUTO mode

ON ERROR GOTO - - need to be able to pass error # or type
ERROR (<dummy-arg>) →

DRAW [x1, y1] TO x2, y2 - plot a line ALSO need an UNDRAW

CHAIN / TRACE $\frac{OFF}{ON}$ / APPEND (with auto-REN?) / BLINK $\frac{ON}{OFF}$

LOAD: <filename>[, $\frac{B}{P}$] - loads a non-tokenized file (ASM-603)
SAVE: saves " " " " " "

WHILE <cond> GOSUB <line#> GOSUB <line#> UNTIL <cond>

What will a 'STEP ON' do after a 'TRACE ON' is executed?

New Reserved Words Syntax

Caution - New BASIC tokens will be incompatible with OLD
 at end, do an automatic CTL-R and then make WRPOS be where edit cursor is, then print a CR.

- ✓ EDIT ^{line-number or} expression needs at end fix and HT fix
- ✓ POS (string-name, string-name)
- ✓ AUTO [starting-line-number [increment]] AUTO to start AUTO Lns CTL-E or ESC? to stop
- ✓ ON ERROR GOTO line-number
- ✓ CHAIN filename [B/P]

✓ MUSIC or NOTE note, octave, duration ~~note, octave, attack, sustain, release~~

- ~ TRACE ON trace execution
- ~ TRACE OFF
- ~ APPEND filename [B/P]
- ✓ TOD(0) syntax like time, but returns HHMMSS
- ✓ BLINK ON cursor blinking turned on
- ✓ BLINK OFF " " turned off

- DRAW $x_1, y_1 \overset{TO}{\rightarrow} x_2, y_2$ draw a line from x_1, y_1 to x_2, y_2
- ~ XREF LINE xref GOTOs, GOSUBs
- XREF VAR1 xref variable names
- ✓ BYE exit BASIC to monitor / reset RTC BLINK TOD

WHILE condition GOSUB line-number
 OPEN ^{INPUT} / ^{OUTPUT} filename [B/P] asks for ^{normal} cassette setup then ^{sets up} USART

CLOSE filename ^{writes on EOF if open}
 GET filename, variable list; ^{ON FILE} ERROR GOTO line#; AT END GOTO line#
 PUT filename, {REM="comment-message" / expression list / END}
 ERROR(0) returns last file error type
 0 = no error
 1 = wrong filename
 2 = checksum error

'INPUT' token MUST be before 'PUT' token in token table
 BASIC startup must ask for current time of day to initialize TOD
 play a note on synthesizer when CTL-G is encountered

New Reserved Words LIST

COMMANDS

EDIT
AUTO
APPEND
XREF
BYE

PARAMETER WORDS

ERROR	ON ERROR ^(also a function) GOTO ^(and for files)
ON	BLINK/TRACE
OFF	BLINK/TRACE
LINE	XREF LINE
VARI	XREF VARI

SBLs

~~ON error goto~~
CHAIN
MUSIC or NOTE
TRACE
BLINK
DRAW
WHILE
OPEN
CLOSE
GET
PUT <sup>(also used as
part of OPEN
OUTPUT)</sup>

ALREADY EXISTING WORDS USED IN NEW WAYS

ON	ON ERROR GOTO
GOSUB	WHILE cond GOSUB in
INPUT	OPEN
OUT	OPEN "OUTPUT"
END	AT END GOTO or END = of GET ^{and} PUT EOF
REM	comment record of PUT

FUNCTIONS

POS
TOD
ERROR

Description & SYNTAX for new COMMANDS, STATEMENTS and FUNCTIONS

[] = optional	{ } = pick only one	<xxx> = replace with appropriate parameter
----------------	-----------------------	--

Commands

XREF {LINE|VARI}
 AUTO [<line#>[,<increment>]]
 APPEND <filename>[, {B|P}]
 SWAP [ON|OFF]
 BYE

cross-references line-numbers or variable-names.
 automatic line numbering when a leading space is typed.
 loads a program after another program, then renumbers.
 allows two programs to be co-resident.
 exits BASIC, resets stack and wormholes, goes to tape mode.

Statements

ERROR GOTO <line#>

goes to the specified line when an error occurs.

MUSIC <numeric-note>,<octave>,<duration>

plays a note via PAIA D/A

WHILE <condition> GOSUB <line#> ^{repeatedly} GOSUBs the specified line while the condition is true

CHAIN <filename>[, B|P]

loads the specified program

TRACE {ON|OFF}

when ON, prints each line before it is executed

BLINK {ON|OFF}

when ON, the cursor blinks

FILE: INP <filename>,<expression-list>; ERROR GOTO <line#>; END GOTO <line#>

inputs from tape the named expressions

FILE: OUT <filename>,{<expression-list>| REM="<comment-message>"}

outputs to tape the expression(s) or a comment record

FILE: OPEN <filename>

sets up data area for the file

FILE: CLOSE <filename>

frees data area

LINE <x1>,<y1>,<x2>,<y2>

draws a line from X1,Y1 to X2,Y2

b8

unused/unimplemented statements

b9

"

"

"

ba

"

"

"

bb

"

"

"

Functions

TOD (<dummy-parameter>)

returns the Time-of-day as an 8 digit # ^{units of a second} #####66

POS (<string-variable>,<string-variable>)

returns the position of the 2nd string in the first string

Changes to old BASIC for New BASIC

J = JUMBO

SBL = Statements-that-can-begin-a-line

Change	26DF	to	J	SBLVECT	point to new SBL Vector Table
	21BE	to	J	CMDVECT	point to new Commands Vector Table
	26DC	to	J	NEWRTN	so new SBLs can get executed
	2406	to	J	TOKENS	point to new Token table
	24F6	to	J	TOKENS	
	4658	to	J	TOKENS	
	3642	to	CPI	0BDH	new token for STR#
	205D	to	J	END	set top of BASIC past new stuff
	2000	to	C3	JSTART	set up TOD and WAKEUP + etc
	21B6	to	C3	JNEWTEST	so new SBLs can get executed

```

NEWTEST CALL 3997H 2E3B CD 97 39
        CPI 0BDH 3E FE BD
        JC 21B9H when A < B 4D DA B9 21
        JMP 21C3H 43 C3 C3 21

NEWRTN  CPI 0BDH 2E46 FE BC
        JNC 21D1H 48 D2 D1 21
        CPI 0BDH 4B FE BD
        JC 21D1H 4D DA D1 21
        LXI D, SBLVECT 5D 11 47 4C
        CALL 399EH 53 CD 9E 39
        SUI 19H 56 D6 19
        JMP 26E4H 58 C3 DE 26
    
```

PUT at 2E3B thru 2FB8

START → 2E5B

MUSIC mc must be almost like PLOT; each needs 3 parameters. Do not use RTC!

Use delay loops.

Correct CHAIN so it detects whether or not CR follows <filename> or B/P

if TOD is implemented, BASIC startup must ask for current time, and the regular time function must then LHL D/SHLD a new area maintained in addition to TOD's BCD area. See below + next page. Also BASIC setup must setup new WAKEUP ISR and put ffffffff at 0CDD - 0CD3.

BLINK, TOD, and TIME must all use same WAKEUP ISR. BLINK ON will set a flag to 1, and BLINK OFF will set it to 0. WAKEUP ISR will only CALL BLINKCURSOR RTN if the flag is not zero. SEE NEXT PAGE

ADD New routine to draw a line between two points

put J2E73 at 4C81

ON = 93
OVT = 94

BLINK ^{2E8} CALL QONOFF 2E73 CD 81 2E
STA BLNKFLG 32 5D 4E
RET A C9
JMP 54 FLAG
TRACE CALL QONOFF
STA TRACFLG — need separate trace printout routine
RET

QONOFF CALL 399E 2E81 CD 9E 39
CPI ON-TOKEN 093H FE 93
JZ ITSON FE 91 2E
CPI OFF-TOKEN 0FDH FE FD
JZ ITSOFF CA 96 2E
JMP 21DIH C3 D1 21
ITSON XRA A \ INR A (Syntax Error) 2E91 AF \ 3C
JMP EXIT (so flag ≠ 0) C3 97 2E
ITSOFF XRA A 2E96 AF
EXIT PUSH PSW 2E97 FS
CALL 3997 CD 97 39
POP PSW FI
RET C9

New BLINK RTN must check if POS ≠ 0 or 1 if so, then no change

WHILE

CALL 39F4?

JZ ———?

CALL 202E

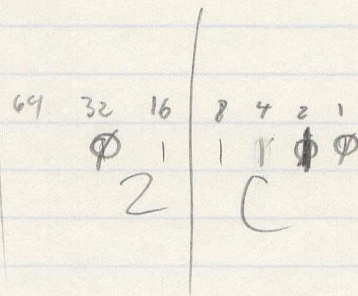
CALL 3922

MVI B, GOSUB token

CALL 398F — make sure next token is GOSUB

24 HR BCD TIME ROUTINE

MVI B,3 ← byte count -1
 LXI H,0C07H LSByte HHmmSS%%
 TIMAGIN MOV A,m
 INR A
 DAA
 MOV MA 4 5 6 7
 CPI 60H
 JC IORET
 MVI M,0
 DCX H
 DCR B
 JNZ TIMAGIN
 MOV A,m
 INR A
 DAA
 MOV MA
 CPI 24H
 JC IORET
 MVI M,0
 JMP IORET



OLD

— Lookup tables —

PROPOSED

ADDR	DW
2E3B	2640
2E3D	25CF
2E3F	45FA
2E41	2518
2E43	2520
2E45	45FB
2E47	2621
2E49	03A9/
2E4B	2549
2E4D	253B
2E4F	4768
2E51	26EF

etc

2E7D 28EE
2E7F on = TOKEN Table

Routine	TOKEN
RUN	A0
LIST	A1
VERIFY	A2
SCR	A3
CLE	A4
LOAD	A5
CON	A6
LINE/EDIT	A7
REN	A8
flp	A9
SAVE	AA
LET	80

etc

EXIT 96

//

ADDR	DW	
2E3B	2640	
2E3D	25CF	
2E3F	45FA	
2E41	2518	
2E43	2520	
2E45	45FB	
2E47	2621	
2E49	EDIT	
2E4B	2549	
2E4D	253B	
2E4F	4768	
2E51	XREF	AB
2E53	AUTO	AC
2E55	APPEND	AD
2E57	SWAP	AE
2E59	UNUSED (BYE)	AF
2E5B	LET	80

etc

2E87	EXIT	28EE	96
2E89	XXXXXXXXXX		
2E8A	ERROR	B0	
	MUSIC	B1	
	WHILE	B2	
	CHAIN	B3	
	FILE	B4	
	TRACE	B5	
	BLINK	B6	
	UNUSED	B7	
		B8	
		B9	
		BA	
		BB	

chg 26AB POS
to JNC NEWRTN
NEWRTN will check out new
sketches

2E99
2E9B

NEWRTN CPI 0BCH
JNC 21D1H syntax error
CPI 0BDH if AZECH
JC 21D1H if A < BPH
SBI 19H so B0 beca 97H
JMP 26DE for table

Chg 29DF to JSBLVECT
" 21BE "JCMOVECT

LXI D, SBLVECT
CALL 399E

VAL thru STEP
will need new TOKENS
and the corresponding CPI or
changed

FIND out how
'OUT' works
does it check for Crea 1?
where does it go to when
done?

New Token Table

~~pages - change~~
~~2406H to 49A0H~~
~~24F6H to 49B0H~~
~~4658H to 49B0H~~
~~3642H to FE7B8~~

~~VAL thru STEP will need~~
~~new CBI xx in their routines~~
~~due to new token values~~
~~Also, check compress/expand routine's use~~
~~of EXIT, THEN, and ELSE to precede~~
~~line #3 in table, 97H~~

9 ***

49B0 DB 080H, 'LET'

etc

xxxx

DB 096H, 'EXIT'

~~0B0H, 'STEP'~~~~0B1H, 'TO'~~~~0B2H, 'THEN'~~~~0B3H, 'TAB'~~~~0B4H, 'ELSE'~~~~0B5H, 'CUR'~~~~0B6H, 'ASC'~~~~0B7H, 'VAL'~~~~0BDH, 'STR'~~

0A0H, 'RUN'

etc

0AAH, 'SAVE'

0ABH, 'XREF'

0ACH, 'AUTO'

0ADH, 'APPEND'

0AEH, 'SWAP'

0AFH, 'BYE'

0EDH, 'C'

etc

0DFH, 'TIME'

0F8H, 'OPEN'

0F9H, 'CLOSE'

0FAH, 'TOD'

0FBH, 'LIVE'

EDIT replaces LINE

XREF LINE
XREF VARI

AUTO[In, In]

APPEND filename[, B/P]

SWAP ON
SWAP

unused

for FILE

" "

time of day function

for XREF, for XREF LINE versus XREF VARI

DB 0B0H, 'ERROR'

0B1H, 'MUSIC'

0B2H, 'WHILE'

0B3H, 'CHAIN'

0B4H, 'FILE'

0B5H, 'TRACE'

0B6H, 'BLINK'

0B7H, 'LINE'

B8

B9

BA

BB

ERROR GOTO line #

MUSIC note, duration

WHILE cond GOSUB line #

CHAIN filename (B/P)

FILE: OPEN
CLOSE
INP
OUT

draw a line

(unused)

New Token Table

 Φ FCH, 'VARI'

Same as for 'LINE'

 Φ BEH, ~~XXXXXX~~ 'INSTR' Φ FEH, 'x' unused Φ ffH ; end of table Φ FDH, 'OFF'

EDIT

SYNTAX

EDIT line number

FUNCTION

edits a line in the same manner as ASM GØ3's edit function

USES

correct typing errors, add to a line, change line numbers, change wording, etc.

RESTRICTIONS

the line to be edited must already exist

COMMANDS

CTL-S move the edit cursor right

CTL-T move the edit cursor left

CTL-Q move the edit cursor to the beginning of the line

CTL-R move the edit cursor to the end of the line

BACKSPACE erase one character from the line. The character erased is the one preceding the edit cursor

WORDERASE same as backspace, except a full word is erased

CARRIAGE RETURN

~~CTL-E~~ end the edit function, and enter the line

any other non CTL-character (and ~~except~~ HT) will be inserted into the text at the edit cursor's position

* means not implemented yet

EDIT

6000-6246

49BQ-4BF6

syntax: EDIT line-number

error messages: 'Syntax error' when no line number is specified.

'Line number error' when a line does not already exist.

use: for editing a line in BASIC which already exists; very much akin to ASM 603's editor, except only 1 line at a time can be edited.

commands: CTL-S - move edit cursor right

CTL-T - move edit cursor left

RETURN - end the edit function

CTL-Q - move edit cursor to beginning of line

CTL-R - move edit cursor to end of line

backspace - erase one character

CTL-W (word erase) - erase characters until a character that is not A-Z, a-z, or 0-9 is encountered

any other character that is not a CTL character (except HT) is inserted into the line at the cursor position

problems: HTs are not ~~inserted~~ or edited or censored over correctly
an HT will be inserted OK, but will show on screen as a greek char, and can then be censored over correctly, but an HT in a line before editing is displayed right but is not censored over correctly. ~~Cursoring over a previously entered HT will blow up edit.~~

future commands: ~~CTL-P - edit line before current line~~

~~CTL-N - edit line after current line~~

EDIT 'string to be found' - edit the line that has the string in quotes.

2406

24F6

4658

402F

EDIT problems

Junk after line # in EDIT statement should cause an error
HT

(right after param)
call 3BB3 or whatever
to see if next char = CR

~~fix CTL R~~

~~fix insert on EOL~~

fix

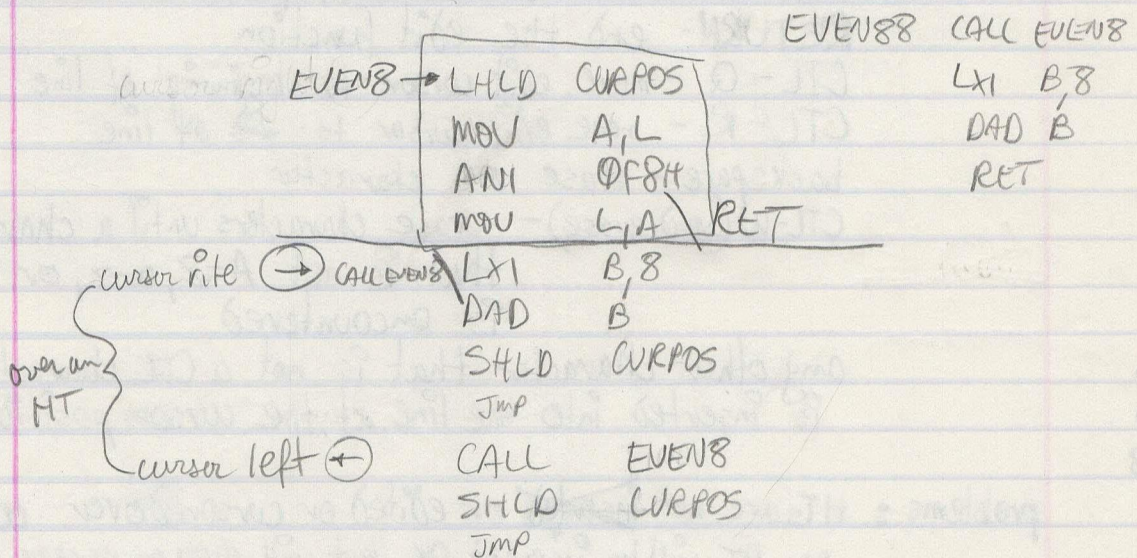
~~HT insertion~~

~~HT cursoring over~~

~~multiple HT delete~~

~~CR for END or ESC when line length > 63~~

~~various intermittent errors~~



EDIT9 4900-4C24
 inclusive
 Ver 0.92 (6000-6274)

L2E49 J4900
 L2F38 A7445444449654
 to E 0 1 T
 L205D J4C25

EDIT	CALL	PARAM (3BB8)
	LHLD	49ACH
* FINAGIN	MOV	B, H
	MOV	C, L
	MOV	A, m
	CPI	01H
	JZ	2201H
	INX	H
	CALL	39E3H
	CALL	CmPR
	JC	2201H
	JZ	FOUND
	LDAX	B
	MOV	L, A
	MVI	H, 0
	DAD	B
*	JMP	FINAGIN

get line # into DE
 get start of text
 put addr of line
 into BC as well as HL
 get line length byte
 check if at end of text
 line # error if at end
 advance to line # DW
 and get it into HL
 is it line # we want?
 if too high
 if we found it
 otherwise get line length
 and put it into HL
 with H=0
 so HL = start of next line
 and try again

FOUND	MOV	H, B
	MOV	L, C
	LXI	D, 48EAH
	CALL	24AFH
	LXI	H, 48EAH
	SHLD	START
	SHLD	BUFFPOS
	XCHG	
	SHED	END
*	CALL	CRDUT
	LXI	B, -40H
	LHLD	0C0EH
*	MVI	m, 07FH
	MOV	
	MOV	
	DAD	B
	SHLD	LINSTR
	SHLD	CURPOS

we found the line, so
 get its addr into HL
 and DE gets addr of uncompressed buffer
 expand the line into buffer (DE = CR at end of line)
 load HL with start of buffer
 and store it at
 the two pointers
 get end of line addr
 into its pointer too
 do a CR past edit line in case of scroll
 so BC = 1 negative line length
 get WH1's cursor pos
 blank it
 and save the real WH1
 position in DE stack
 so HL = edit line 1st char
 save that for edit
 ditto

PUSH H

5 6 7 8 9
 0 1 2 3 4 5 6 7 8 9
 HL DE
 6 9
 END = 8
 POP PTR = 5

SHLD
 MVI
 CALL
 LHL

000EH
 A, 0FFH
 WH1
 START

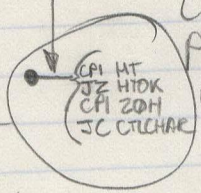
and for WH1 to print the line
 put the edit cursor
 onto the screen
 get start of edit buffer

— need to eliminate any/all CTL chars (except HT) —

don't print on screen
 and don't have in
 buffer

PRTAGIN

MOV
 CPI
 JZ
 CALL
 INX
 JMP



A, m
 CR
 PRTEND
 WH1
 H
 PRTAGIN

get a char
 is it a CR?
 if so, we're done printing
 otherwise print the char
 and incr to next char
 and print some more

CTLCHAR PUSHH XCHG LHL END DCR H SHLD END INX H XCHG CALL MOVLEFT POP H JMP PRTAGIN

PRTEND

LHL
 MVI
 DCR
 SHLD
 POP
 SHLD
 XRA
 DCR
 STA

000EH
 M, 07FH
 H
 LINEND
 H
 000EH
 A
 A

get WH1's cursor position
 plank the cursor
 decr one pos
 and store for edit
 get WH1's real cursor pos
 and restore it
 zero A, then
 make it 0FFH
 and initialize flag

PUT JUST BEFORE INSERT LXI D, 493AH LHL END CALL CMPR JNZ GETIMP LXI B, 23ADH JMP JZ2PH

BEGIN
 GETIMP

CALL
 CPI
 JZ
 CPI
 JZ
 CPI
 JZ
 CPI
 JZ
 CPI
 JZ
 CPI
 JZ
 CPI

WH0
 CTLE
 ENDIT
 BKSPC
 ERASECHR
 WDERASE
 ERASEWD
 HT
 INSERT
 CTLS
 RIGHT
 CTLT
 LEFT
 CTLQ

get a char
 end of edit?
 if so, go to end rtn
 backspace?
 if so, erase a char
 word erase?
 if so, erase a word
 horizontal tab?
 allowable ctl-char to insert
 move cursor right?
 if so
 move cursor left?
 if so
 move cursor to start of line

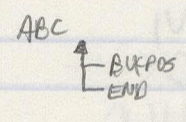
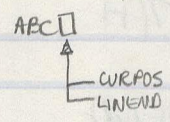
LHL LINEND
 INX H
 SHLD 000EH

55223
55223
55223

WIND 190
CAL 190
M 190
A 190
H 190
START

WIND 190
CAL 190
M 190
A 190
H 190
END

WIND 190
CAL 190
M 190
A 190
H 190
END



WIND 190
CAL 190
M 190
A 190
H 190
END

WIND 190
CAL 190
M 190
A 190
H 190
END

WIND 190
CAL 190
M 190
A 190
H 190
END

WIND 190
CAL 190
M 190
A 190
H 190
END

WIND 190
CAL 190
M 190
A 190
H 190
END

WIND 190
CAL 190
M 190
A 190
H 190
END

WIND 190
CAL 190
M 190
A 190
H 190
END

JZ STRTLIN
CPI CLR
JZ ENDLIN
CPI ZOH
JC BEGIN

if so
move cursor to end of line?
if so
if the character is less than ZOH,
(CH-char) then don't insert it

☆ put ~~overflow~~ time overflow-error checker here / make the CPI HT above jump to here, not INSERT

INSERT

PUSH PSW
LHLD END
XCHG
LHLD BUFPOS
~~~~~  
CALL MOVITE  
LHLD LINEND  
XCHG  
LHLD CURPOS  
CALL MOVITE  
POP PSW  
LHLD BUFPOS  
MOV M, A  
INX H  
SHLD BUFPOS  
LHLD CURPOS  
CPI HT  
; JZ HTRTN \*\*not implemented yet\*\*  
ORI ZOH  
MOV M, A  
INX H  
SHLD CURPOS  
LHLD END  
INX H  
SHLD END  
LHLD LINEND  
INX H  
SHLD LINEND  
JMP BEGIN

save input char

OK → ~~get addr of end of buffer~~

OK → ~~into DE~~

OK → ~~and addr of position in buffer~~

~~see if equal~~

move everything over (at least the CR)

get screen end of  
line into DE

and pos in line into HL

then move screen line (from cursor on) rite

get input char

get where to put it  
and insert it

but keep ptr at same old char  
and save it

get screen insert addr  
is it a HT?

if so, special treatment needed  
otherwise, make char scrn compatible  
and insert it onto screen

and adjust cursor BW  
by saving it

adjust end  
of buffer

pointer

adjust end  
of screen

line pointer

and get more input



HTRTN

LHLD END

INX H

SHLD END

HTAGIN

LHLD LINEND

INX H

SHLD LINEND

LHLD CURPOS

XCHG

CALL EVEN88

~~CALL~~  
~~CALL~~  
~~CALL~~  
~~CALL~~

CALL CMPR

JZ BEGIN

LHLD LINEND

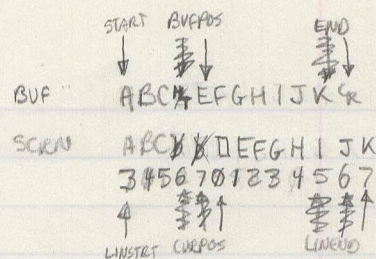
XCHG

LHLD CURPOS

CALL MODRITE

JMP HTAGIN

(MVI M, 07FH  
INX H  
SHLD CURPOS





|         |                                                                                                 |                                                                                                              |                                                                                                                                                                                                                                                                                        |
|---------|-------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ERASCHR | CALL<br>JMP                                                                                     | ERASONE<br>BEGIN                                                                                             | erase one char. if possible<br>the get more input                                                                                                                                                                                                                                      |
| ERASONE | LHLD<br>XCHG<br>LHLD<br>CALL<br>* RZ<br>DCX<br>MOV<br>CPI<br>JNZ<br>XRA<br>STA                  | START<br><br>BUFFPOS<br>CMAR<br><br>H<br>A, m<br>HT<br>NOTAB<br>A<br>TABFLAG                                 | see if<br>at the start<br>of the<br>line?<br>if so, can't backspace<br>get char we are deleting.<br>is it a HT?<br>if not, otherwise<br>zero A<br>so we can set flag                                                                                                                   |
| NOTAB   | LHLD<br>XCHG<br>LHLD<br>* CALL<br>LHLD<br>DCX<br>SHLD<br>LHLD<br>DCX<br>SHLD                    | END<br><br>BUFFPOS<br>MOVLEFT<br>END<br>H<br>END<br>BUFFPOS<br>H<br>BUFFPOS                                  | get end of buffer<br>into DE and pos<br>in buffer into HL<br>and move every thing (from cursor on) left<br>adjust<br>buffer end<br>pointer<br>adjust<br>buffer pos<br>pointer                                                                                                          |
| AGINTAB | LHLD<br>XCHG<br>LHLD<br>CALL<br>LHLD<br>mvi<br>DCX<br>SHLD<br>LHLD<br>DCX<br>SHLD<br>LDA<br>ORA | LINEND<br><br>CURPOS<br>MOVLEFT<br>LINEND<br>m, 07FH<br>H<br>LINEND<br>CURPOS<br>H<br>CURPOS<br>TABFLAG<br>A | get end of screen line<br>into DE<br>and cursor position into HL<br>then move all that over 1 (left)<br>get end of screen line again<br>and blank old last char<br>then adjust<br>the pointer<br>adjust<br>screen cursor<br>pointer<br>get tab flag to see<br>if we are deleting a TAB |

Decr to char  
we are deleting



```

RNZ
LHLD CURPOS
DCX H
MOV A,m
CPI SCRNSPC
JNZ TABEND END
TABEND XRA A
      DCR A
      STA TABFLAG
      RET
  
```

return if not deleting a tab  
 otherwise get cursor position  
 decr to char before it  
 get the char so we can see  
 if a scrn space  
~~then delete it maybe~~ if not  
 otherwise zero A  
 then make it 0FFH  
 to reset the TAB FLAG  
 and RET cause were done

```

RIGHT LHLD LINEND
      XCHG
      LHLD CURPOS
      CALL CMPR
      JZ BEGIN
      XCHG LHLD BUFPPOS MOV A,m CPI HT JZ RIGHT XCHG
      INX H
      SHLD CURPOS
      MOV A,m
      MVI m,0FFH
      DCX H
      MOV m,A
      LHLD BUFPPOS
      INX H
      SHLD BUFPPOS
      JMP BEGIN
  
```

check if we are at end of line  
 if we are can't go RIGHT any farther  
 so HL = CURPOS + 1  
 store new position  
 get screen character  
 and put a cursor where char was  
 then decr to where cursor used to be  
 and put the character there  
 adjust the buffer pointer  
 then get more input

```

LEFT LHLD LINSTRT
      XCHG
      LHLD CURPOS
      CALL CMPR
      JZ BEGIN
      XCHG LHLD BUFPPOS DCX H MOV A,m CPI HT JZ RIGHT XCHG
      DCX H
      SHLD CURPOS
      MOV A,m
      MVI m,0FFH
  
```

check if we are at beginning of the line  
 if we are, can't go left any farther  
 so HL = CURPOS - 1  
 store new position  
 get the screen character there  
 and put a cursor where char was



HRIGHT

~~CALL EVEN8~~  
MVI M, 07FH  
LHLD LINEND  
XCHG  
CALL EVEN8  
SHLD CURPOS  
CALL MOVERITE  
LHLD CURPOS  
MVI M, 0FFH  
JMP HREND

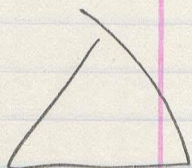
HLEFT

~~MVI M, 07FH~~  
~~INX H~~  
XCHG  
LHLD LINEND  
XCHG  
CALL MOVELEFT  
CALL EVEN8  
SHLD CURPOS  
MVI M, 0FFH  
JMP HLEND

blank cursor

Increment to char after cursor

HL = curpos, DE = linen



CALL EVEN8  
XCHG  
LHLD CURPOS  
CALL CMPR  
JNZ AGINTAB



|         |      |          |                                   |
|---------|------|----------|-----------------------------------|
|         | INX  | H        | then incr to where cursor was     |
|         | MOV  | M, A     | and put character there           |
| HLEND   | LHLD | BUFFPOS  | adjust                            |
|         | DCX  | H        | the buffer                        |
|         | SHLD | BUFFPOS  | pointer                           |
|         | JMP  | BEGIN    | then get more input               |
| ERASEWD | CALL | ERASEONE | erase one char                    |
| ★       | JZ   | BEGIN    | zero flag set if at start of line |
|         | LHLD | BUFFPOS  | get HL = position in buffer       |
| ★       | DCX  | H        | decr to character we might delete |
|         | MOV  | A, M     | get that char                     |
| ★       | CALL | ALPHNUM  | see if A-Z, a-z, 0-9              |
| ★       | JZ   | ERASEWD  | erase more if it was              |
| ★       | JMP  | BEGIN    | else get more input               |
| CMPR    | MOV  | A, D     | put D into A                      |
|         | CMP  | H        | compare with H                    |
|         | RNZ  |          | if not equal with flags set       |
|         | MOV  | A, E     | put E into A if H equaled D       |
|         | CMP  | L        | compare with L                    |
|         | RET  |          | then RET with flags set           |
| MOVRITE | MOV  | B, D     | put top                           |
|         | MOV  | C, E     | DE = top, HL = bottom             |
|         | INX  | B        | into BC                           |
| RITE    | LDA  | D        | incr to dest byte                 |
|         | STAX | B        | get source byte                   |
|         | CALL | CMPR     | and store it at dest byte         |
|         | RZ   |          | see if we've reached bottom yet   |
|         | DCX  | D        | if we got there                   |
|         | DCX  | B        | otherwise decr source ptr         |
|         | JMP  | RITE     | and decr dest ptr                 |
| MOVLEFT | MOV  | B, H     | then move more                    |
|         |      |          | put bottom                        |
|         |      |          | DE = top, HL = bottom             |



|       |      |       |                              |
|-------|------|-------|------------------------------|
|       | MOV  | C,L   | into BC                      |
|       | DCX  | B     | decr to dest byte            |
| LEFT1 | MOV  | A,m   | get the source byte          |
|       | STAX | B     | and store it at dest byte    |
|       | CALL | CMPR  | see if we've reached top yet |
|       | RZ   |       | if we got there              |
|       | INX  | H     | otherwise incr source ptr    |
|       | INX  | B     | and incr dest ptr            |
|       | JMP  | LEFT1 | then move some more          |

(4B98)  
ENDIT

NOP

JMP

2193H

for future use

just past the CR after a line of input routine

→ LHL CURPOS \ XCHG \ LHL LINSTRT \ CALL CMPR \ JZ BEGIN

STRTLIN

LHL

START

make buf pos

SHL

BUFPOS

equal start of line

LHL

CURPOS

get cursor

DCX

H

position -1

XCHG

into DE

LHL

LINSTRT

get start into HL

SHL

CURPOS

and make it new cursor position

CALL

MOVITE

then move every thing over

LHL

CURPOS

then put a cursor

MVI

M, 0FFH

symbol at Curpos

JMP

BEGIN

and then get more input

(4BC4)

ENDLINE

; see next page

TABFLAG

DS

1

0FFH when not deleting a TAB, 00 when deleting

LINSTRT

DS

2

start of line on screen pointer

CURPOS

DS

2

position of edit cursor pointer

LINEND

DS

2

end of screen-line pointer

START

DS

2

start of buffer-line pointer

BUFPOS

DS

2

position in buffer pointer

END

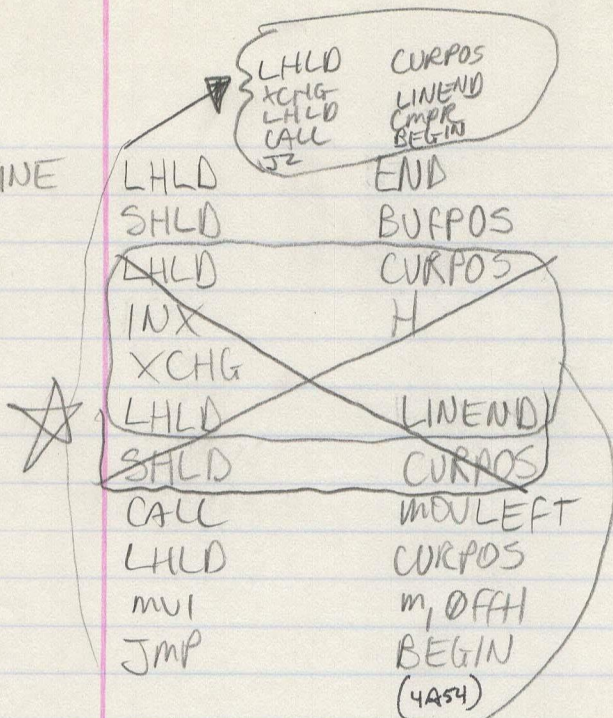
DS

2

end of buffer-line pointer



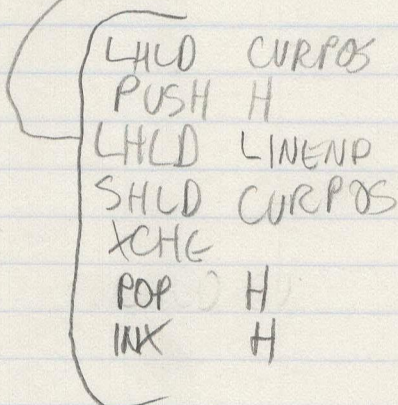
ENDLINE



make bufpos

equal to start of line  
get cursor position  
position + 1  
into DE

get end into HL  
and make it new cursor position  
then move it all over  
then put a cursor  
symbol at Curpos  
and then get more input



DE = TOP } for MOVELEFT  
HL = BOT }

EVEN8 LHL D CURPOS

```

    MOV A, L
    ANI 0FH
    MOV L, A
    RET
  
```

EVEN8 CALL EVEN8

```

    LXI B, 8
    DAD B
    RET
  
```



BH = JNC

BL = JC

9

Changes - make the list part (CALLS WH1) skip over any CTL-char except HT in a line

make CTL-W erase any char = A-Z, a-z, 0-9

CALL ALPHNUM to check 0-9, A-Z, a-z; call ALPH to check A-Z, a-z; call LOWER to check a-z  
alpha numeric checker - zero flag set if char is A-Z, a-z, 0-9

|         |            |                                      |
|---------|------------|--------------------------------------|
|         | PUSH B     | save B                               |
|         | MOV B, A   | save A                               |
| ALPHNUM | CPI '0'    |                                      |
|         | JC NOT     | if a CTL-char or less than '0' (30H) |
|         | CPI '9'+1  |                                      |
|         | JC OK      | if '0'-'9'                           |
| ALPHA1  | CPI 'A'    |                                      |
|         | JC NOT     | if >'9' but <'A'                     |
|         | CPI 'Z'+1  |                                      |
|         | JC OK      | if 'A'-'Z'                           |
| LOWER1  | CPI 'a'+1  |                                      |
|         | JC NOT     | if >'Z' but <'a'                     |
|         | CPI 'z'+1  |                                      |
|         | JC OK      | if 'a'-'z'                           |
| NOT     | XRA A      | set zero flag                        |
|         | INR A      | then unset it                        |
|         | JMP DONE   |                                      |
| OK      | XRA A      | set zero flag                        |
| DONE    | MOV A, B   | restore A                            |
|         | POP B      | restore B                            |
|         | RET        | and RET                              |
| ALPHA   | PUSH B     | save B                               |
|         | MOV B, A   | save A                               |
|         | JMP ALPHA1 | go do it                             |
| LOWER   | PUSH B     | save B                               |
|         | MOV B, A   | save A                               |
|         | JMP LOWER1 | go do it                             |



# POS function for BASIC

A = POS(B\$, A\$)

POS (ASM)

0-6

POS2 0-7

returns the <sup>1st</sup> position of A\$ in B\$ or  
returns a 0 if A\$ is not in B\$

~~RETURN 0~~ RETURN 0 if B\$ or A\$'s len = 0

POS CALL 3985H

CALL 3724H

JNZ 21F5H

PUSH H

XCHG

SHLD BLEN

CALL 3980H

CALL 3724H

JNZ 21F5H

XCHG

SHLD ALEN

POP H

LXI B, 1

AGAIN1

PUSH B

PUSH D

PUSH H

LHLD ALEN

SHLD ASAVE

LHLD BLEN

SHLD BSAVE

AGAIN2

POP H

PUSH H

AGAIN2

LDAX D

CMP M

JNZ NEXTBCHR

PUSH H

LHLD ALEN

DCX H

SHLD ALEN

MOV AH

ORA L

POP H

check for '('

get addr and len of B\$

type error if not a string

save addr of B\$

save length of B\$

check for ;

get addr and len of A\$

type error if not a string

save length of A\$

HL = addr of B\$, DE = addr of A\$

B\$ position pointer

ASMBE 4C28

ALEN = 46A8 4CB0

BLEN = 46A8 4CB2

ASAVE = 46A8 4CB4

BSAVE = 46A8 4CB6

ad = 4CB0 4CB8

L4C27 FF for debugging

ASC vector at 207B = L207B

change to 4C27 = J4C27

ASC reserved word at 2F0B = L2F0B

change to POS 50 4F 53

HIS  
ABCAEF  
↑  
Q7



$BLEN = 2$   $BSAVE = 2$   
 $BC(BH POS PTR) = 9$   
 $ALLEN = 0$   $ASAVE = 1$

|       |      |        |
|-------|------|--------|
| EMPTY | CALL | 39B0H  |
|       | CALL | 3724H  |
|       | JNZ  | 21F5H  |
|       | JMP  | EMPTY1 |

we get here when  $BA = "$   
 " so we need to scan past  
 "A#)"

|      |       |
|------|-------|
| LHLD | BSAVE |
| DCX  | H     |
| SHLD | BLEND |
| XCHG |       |

LAI H, ALEN

21 B0 4C

CHILD ALLEN 390BH  
 CALL CMR HL<sup>(AW)</sup> WITH DE / ALLEN WITH BLEN  
 JC NOT FOUND if BLEN < ALLEN  
 POP H

|     |        |
|-----|--------|
| POP | H      |
| POP | D      |
| POP | B      |
| INX | B      |
| INX | H      |
| JMP | AGAIN1 |

|     |   |
|-----|---|
| POP | 5 |
| POP | H |
| POP | D |
| POP | B |
| LXI | H |
| JMP | 3 |

EMPTY2  
EMPTY1

|     |       |
|-----|-------|
| LXI | H, D  |
| JMP | 3005H |

|       |    |   |
|-------|----|---|
| ALEN  | DS | 2 |
| BLEN  | DS | 2 |
| ASAVE | DS | 2 |
| BSAVE | DS | 2 |

$$BC(BH \text{ POS } P+R) = 9$$

~~POP H, POP D, POP B  
 MOV H, B  
 MOV L, C  
 JMP 30D5H~~

- LHLB BLN  
 MAY 11  
 formerly POPA  
 formerly POP B, but all  
 EC in H

FOUND

~~2A 00 02~~  
~~CALL EBH~~

~~06 00 00 28~~  
~~DEC ? H?~~

~~00 00 00 10~~  
~~MOV AL, 10~~

~~00 00 00 00~~  
~~PRA~~

~~00 00 00 00~~  
~~MOV AL, 00~~

POP B

POP D

POP H

JMP 3005H

med



HL →  
 B# = "QUADRA"  
 B = 6 5 4 3 2

DE →  
 A# = "TEX"  
 A = 3

not found not found

HL →  
 B# = "QUADRA"  
 B = 6 5 4 3

DE →  
 A# = "UAT"  
 A = 3 2 1

HL →  
 B# = "QUAD"  
 B = 4 3 2

DE →  
 A# = "ADR"  
 A = 3

not found

HL →  
 B# = "QUADRA"  
 B = 6 5 4 3

DE →  
 A# = "UAD"  
 A = 3 2 1 0

MOV A, M  
 MOV H, M  
 MOV L, A



## AUTO

### SYNTAX

AUTO starting line-number, optional increment  
the syntax is much like RENUMBER's syntax.

### FUNCTION

allows line numbers to be assigned automatically, with any increment between the line numbers

### USES

decreased amount of typing needed, due to no need to input the line numbers

### RESTRICTIONS

a line number overflow error will occur any time the next AUTO line number would be greater than 65535

~~To initialize the starting line number and increment, use the AUTO command. Defaults to 10,10.~~

~~To use the AUTO-line numbering feature, merely start every line that you want AUTO numbered with a space, and the space will be replaced by the AUTO line-number.~~

To use the AUTO command, just type in  
AUTO with or without the optional other  
numbers. It will stay in AUTO mode  
until you press ~~Ctrl-C~~ **ESC**



## ERROR

### SYNTAX

line number <sup>ON</sup> ERROR GOTO line number ~~CR~~

### FUNCTION

when an ERROR statement is encountered, a flag is set so that when an error occurs, instead of printing an error message and halting, execution will continue as directed in the GOTO statement of the ERROR line. If another ERROR statement is executed after the first, then the new ERROR line supersedes the old one.

### USES

For error catching without an irrecoverable halt. a good use is recovery from a "Line length error" when INPUTting.

### RESTRICTIONS

An irrecoverable "Line number error" will result if the ERROR line goes to itself; i.e. -

40 ERROR GOTO 40

If an ERROR statement is typed while in the direct mode of BASIC, an error may be caused by a future statement.

The ERROR statement does not function for these two errors - "Input error - retype"  
- "Double def error".

After a program with an ERROR statement ends, the first error in direct mode will clear the error flag but will print "Illegal direct error".

### NOTE

It is recommended that you test a program without any <sup>ON</sup> ERROR statements first to get rid of SYNTAX errors.



# ERROR MC

\* MUST PUT C3/03/4A at 2225-2227 \*

|      |          |        |            |                        |
|------|----------|--------|------------|------------------------|
| 49B0 | 3E 00    | ERR    | MVI A, CR  |                        |
|      | 2A B4 48 |        | LHLD 48B4  |                        |
|      | 2B       |        | DCX H      | to error token         |
| 49B6 | 2B       | NOTCR  | DCH H      |                        |
|      | BE       |        | CMP M      |                        |
|      | C2 B6 49 |        | JNZ NOTCR  |                        |
|      | 2B       |        | DCX H      | } in case length = 00H |
|      | BE       |        | CMP M      |                        |
|      | CA C1 49 |        | JZ LENGTH  |                        |
|      | 23       |        | INX H      |                        |
| 49C1 | 23       | LENGTH | INX H      | to line length         |
|      | 23       |        | INX H      | to line number         |
|      | CD E3 39 |        | CALL 39E3  | get line #             |
|      | EB       |        | XCHG       | into DE                |
|      | CD 9E 39 |        | CALL 399E  | incr past ERROR token  |
|      | FE 88    |        | CPI GOTO   |                        |
|      | CA D6 49 |        | JZ NOSYN   |                        |
|      | 2B       |        | DCX H      | error - no goto found  |
|      | 22 B4 48 |        | SHLD 48B4  |                        |
|      | C3 D1 21 |        | JMP SYNERR | go to ! "Syntax error" |
| 49D6 | 22 00 4A | NOSYN  | SHLD TEMP  |                        |
|      | CD 9E 39 |        | CALL 399E  | incr past GOTO         |
|      | FE 97    |        | CPI 97     | Line number?           |
|      | C2 01 22 |        | JNZ LINERR | Line # error if not    |
|      | CD E3 39 |        | CALL 39E3  | get line # into HL     |
|      | 70       |        | MOV A, L   |                        |



BB  
 C2 EE 49  
 7C  
 BA  
 CA 01 22  
 49EE 3E 01  
 32 02 4A  
 3E 8F  
 C3 DA 29

CMP E  
 JNZ LNOK  
 MOV A, H  
 CMP D  
 JZ LINERR *if Line# = goto Line#*  
 MVI A, 1  
 STA ERRFLAG  
 MVI A, REM *disregard rest of line*  
 JMP 290A *then continue execution*

4A00 DS  
 4A02

TEMP DS 2  
 ERRFLAG DB 0

4A03 3A 02 4A  
 B7  
 C2 - 4A  
 3A A8 48  
 C3 28 22  
 2A 00 4A  
 22 B4 48  
 AF  
 32 02 4A  
 C3 AB 28

ERROR LDA ERRFLAG  
 ORA A  
 JNZ OK  
 LDA 48A8  
 JMP 2228  
 OK LHL D TEMP  
 SHLD 48B4  
 XRA A  
 STA ERRFLAG  
 JMP GOTO-RTN

need to fiddle with 382C-3832 to set errflag to 0 when  
 at end of program. (p2 of typewriter page) *make 3832 a JMP?*



## CHAIN

### SYNTAX

line number CHAIN filename, <sup>B</sup><sub>P</sub> optional CR

### FUNCTION

Acts the same as a LOAD except it is to be used within a program.

### USES

For a large program bigger than available memory a good use is a file menu where a user can pick which file he wishes to use.

### RESTRICTIONS

The same restrictions as used for LOAD. Also no variables are passed, but values POKE'd into free memory are OK.

### NOTE

Remember that, like a LOAD, a CHAIN statement will SCRatch the present program to make room for the LOADING program.



CHAIN replaces LET

- 1 replace LET reserved word with CHAIN
- 2 replace LET DW with DW address of CHAIN routine (below).
- 3 install in some free RAM the routine below:

|       |              |                    |
|-------|--------------|--------------------|
| CHAIN | LHLD 48B4H   |                    |
|       | LXI D, 48C3H |                    |
|       | XCHG         |                    |
|       | SHLD 48B4H   |                    |
| NEXT  | LDAX D       |                    |
|       | MOV M, A     |                    |
|       | CPI CR       |                    |
|       | JZ LOAD      | (BASIC's Load Rtn) |
|       | INX H        |                    |
|       | INX D        |                    |
|       | JMP NEXT     |                    |



# MUSIC<sup>numeric</sup>

## SYNTAX

Line number MUSIC note ~~string~~ <sup>numeric</sup> variable or constant, octave numeric variable or constant, duration numeric variable or constant

## FUNCTION

outputs the specified note in the octave for the duration specified, to the PAIA synthesizer.

NOTE range: ~~"A", "A#", ..., "G", "G#"~~ 0 = rest, 1 = C, 2 = C# or Db, etc thru 12

OCTAVE range: 1 thru 5

DURATION range: ~~8 = eighth note~~ 0 = eighth note in 60ths of a second  
4 = fourth note  
2 = half note  
1 = whole note  
~~error if 0~~

## USES

for composition playing by way on INPUT or READ.  
sound effects, etc.

## RESTRICTIONS

the variables and/or constants must be within the ranges listed in the function section

duration does not specify the actual duration; the pulse trigger bit, B7, is on for a 60th of a second, then the step trigger, B6, is on for the specified duration. Then, for 5/60ths of a second, the note is held in the D/A for the decay portion of ADSR but B6 (step trigger) is off. Then a 0 is output to the A/D, to end and the next statement is executed



# MUSIC

note, octave, duration

Rest should jump to duration  
+ skip all else in between

note: 0 = rest, 1-12 = C thru B

octave: 1-5

duration: in 60<sup>th</sup>s of a second

|       |               |       |                      |                                     |
|-------|---------------|-------|----------------------|-------------------------------------|
| 49B0  | CD B8 3B      | MUSIC | CALL PARAM           | get note into DE and A              |
|       | FE 00         |       | CPI 13 <sub>10</sub> |                                     |
|       | D2 DD 21      |       | JNC BadArgError      | if note ≥ 13                        |
|       | F5            |       | PUSH PSW             | save note                           |
|       | CD B0 39      |       | CALL FindTheComma    |                                     |
|       | CD B8 3B      |       | CALL PARAM           | to get octave into DE and A         |
|       | FE 06         |       | CPI 6                | ORA A<br>JZ BadArgError             |
|       | D2 DD 21      |       | JNC BadArgError      | if octave ≥ 6                       |
| 49C4  | F1            |       | POP PSW              | get note into A                     |
| 49C5  | <del>47</del> |       | <del>MOV B, A</del>  | we're going to multiply note by oct |
| 49C6  | 1D CD 49      |       | MULT DCR E           |                                     |
|       | CA CE 49      |       | JZ MULTDONE          |                                     |
|       | 80 C5 49      |       | ADD B                |                                     |
|       | C3 C6 49      |       | JMP MULT             |                                     |
| 49CE  | F6 40         |       | ORI 40H              | make bit 6 hi (A0SR)                |
|       | F5            |       | PUSH PSW             | save it                             |
|       | F6 80         |       | ORI 80H              | make bit 7 hi (pulse)               |
|       | D3 FF         |       | OUT PAIA             |                                     |
| *49D8 | 21 FF FF      |       | LXI H, -1            |                                     |
|       | CD FD 49      |       | CALL SETNWAIT        |                                     |
|       | CD B0 39      | DURAT | CALL FindTheComma    |                                     |
|       | CD B8 3B      |       | CALL PARAM           | get duration into DE                |
| 49E1  | 2F            |       | CMA                  |                                     |
|       | 6F            |       | MOV L, A             | Get Z's comp of duration            |
|       | 7A            |       | MOV A, D             |                                     |
|       | 2F            |       | CMA                  |                                     |
|       | 67            |       | MOV H, A             |                                     |
|       | 23            |       | INX H                |                                     |
|       | F1            |       | POP PSW              |                                     |
|       | D3 FF         |       | OUT PAIA             |                                     |
| *49EA | CD FD 49      |       | CALL SETNWAIT        |                                     |
|       | EE 40         |       | XRI 40H              | turn off bit 6 (A0SR)               |
|       | D3 FF         |       | OUT PAIA             |                                     |
| *49F4 | 21 ~FA FF     |       | LXI H, -5            | for delay                           |
|       | CD FD 49      |       | CALL SETNWAIT        |                                     |
|       | AF            |       | XRA A                |                                     |



## PROPOSED CHG TO MUSIC

SEND PULSE TRIG + ETC TO PWA  
SET UP TIMER, BUT DON'T WAIT YET  
~~SEND~~ DO COMPUTATIONS FOR NOTE + DURATION  
WAIT FOR PULSE TO END  
SEND OUT NOTE  
DO COMPUTATIONS FOR DECAY  
WAIT FOR END OF NOTE  
SEND DECAY

IN OTHER WORDS, DO COMPUTATIONS FOR  
NEXT STEP BEFORE CHECKING FOR  
END OF CURRENT STEP

ALSO, HAVE "WAIT FOR END OF DURATION OR ETC" routine  
also check for an ESC  
especially duration's end checker

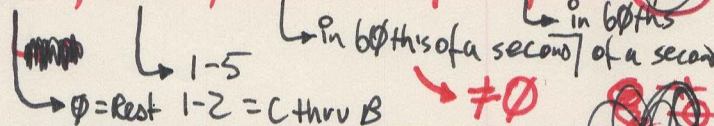


POSSIBLE?  
New Syntax



MUSIC note, octave, sustain time, decay time

sustain + decay times  
cannot equal 0



03 FF  
C3 DA 29  
49FD F3  
22 00 0C  
21 FF FF  
22 02 0C  
FB  
FS  
4A09 3A 03 0C  
B7  
C2 09 4A  
F1  
C9  
4A12 06 0C

SETNWAIT

OUT PAIA  
JMP exec next statement  
DI  
SHLD TIMER  
LXI H, 0FFFFH  
SHLD TIMER+2  
EI  
PUSH PSW  
WAIT LDA TIMER+3  
ORA A  
JNZ WAIT  
POP PSW  
RET  
MVI B, 12

SETUP

Put MUSIC at 2E80-2E84 if ERROR was  
4D 55 53 49 43 previously installed; otherwise put mus  
Put J 49B0 at 2E51

ALSO CHG 2225-2227 to 3A A8 48 if error was installed

octave -1 = how many times we add 12 to note



Need to make delays smaller  
so we can play faster  
notes

currently decay delay is  $\frac{1}{12}$  of a second

It is the main slow down

\* ADD a decay duration? \* and/or make decay a function of WAKEUP

DECAY HANDLER WILL  
SHLD DECAY AT TIMER  
SHLD EFF AT TIMER+2  
LXI H, 0FFFFH  
SHLD WAKEUP JMP 2900  
where waking will out PAIA, phi, then DISCONNECT ITSELF FROM WAKEUP i.e. LXI H, 10RET SHLD WAKEUP



# ~~FLIP~~ SWAP

have 'flip' change from a program at 4B00 - 6FFF

to the program at 6000 to 7BFF

swap 495F DW with 4AF8  
swap 48AA DW with 4AFA  
swap 49AC DW with 4AFC  
swap 49AE DW with 4AFE

LHLD 49AC  
XCHG  
LHLD 4AFC  
SHLD 49AC  
XCHG  
SHLD 4AFC  
LHLD 48AA  
XCHG  
LHLD 4AFA  
SHLD 48AA  
XCHG  
SHLD 4AFA  
LHLD 49AE  
XCHG  
LHLD 4AFE  
SHLD 49AE  
XCHG  
SHLD 4AFE

} switch start of test

} switch top of test

} switch top of memory



aj  
^  
2 b q ps

SWAP

~~FLIP~~ - switch between 2 programs resident in BASIC

CHG DW at 2E49 from J3A9 to J49B0 (or 49B0)  
CHG 2F38-2F3C from <junk> to A7B46B4CB49B  
and CHG 49B0 on to:

|      |    |    |     |                                                          |
|------|----|----|-----|----------------------------------------------------------|
| 49B0 | 00 |    | NOP | for an 0FFH when testing                                 |
| 49B1 | 2A | 5D | 20  | LHLD 205DH DW containing end of BASIC                    |
| 49B4 | EB |    |     | XCHG                                                     |
| 49B5 | 2A | -- | 49  | LHLD SAVEBOT                                             |
| 49B8 | 22 | 5D | 20  | SHLD 205DH DW 10                                         |
| 49BB | EB |    |     | XCHG                                                     |
| 49BC | 22 | -- | 49  | SHLD SAVEBOT                                             |
| 49BF | 2A | 60 | 20  | LHLD 2060H DW containing start of Top of mem search addr |
| 49C2 | EB |    |     | XCHG                                                     |
| 49C3 | 2A | -- | 49  | LHLD SAVETOP                                             |
| 49C6 | 22 | 60 | 20  | SHLD 2060H                                               |
| 49C9 | EB |    |     | XCHG                                                     |
| 49CA | 22 | -- | 49  | SHLD SAVETOP                                             |
| 49CD | C3 | 03 | 20  | JMP 2003H Warm start address                             |
| 4    |    |    |     | end of code                                              |
| 49D0 | 00 | 60 |     | SAVEBOT DW 6000H                                         |
| 49D2 | FF | 7B |     | SAVETOP DW 7BFFH                                         |

only problem is the first time both partitions should be executed with SPJ2000 in order to initialize each partition.  
to fix that - MAKE 2000 jump to: LXI H, REO REO MVI A, C9 STA 49C0 CALL FLIP MVI A, C3 STA 49CD JMP 2041



## PUT

### SYNTAX

linenumber PUT filename, string or numeric variable or constant,

### FUNCTION

puts string or numeric data onto a cassette file called "filename" in Polyphase.

### USES

Save data like a disk system

### RESTRICTIONS

Only one string or number can be saved at a time, and it is recorded in Polyphase.

## GET

### SYNTAX

linenumber GET filename, string or numeric variable.

### FUNCTION

gets string or numeric data from a cassette data file named "filename" in Polyphase.

### USES

Save data and the GET it like a disk system.

### RESTRICTIONS

Only one string or number can be loaded at a time, and it is recorded in Polyphase.



to CHG LET to GET

pat 47 at 2E8D

put J49CB at 2E51

↑ or where ever GET routine is at

to CHG PRINT to PVT/@

put 55 2E89

54                      ZF8A

92 2E8B

21 2F8C

put J 49BØ at 2E 55

or wherever PUT rfn is at

PUT GET Filename, B, P, exp

GET exp can be VAR #'s and strings  
PUT exp can be VAR #'s, strings, constants (2 or "4")  
etc

49D7

LOAD = 45FB (AF)

$$\text{SAVE} = 4768$$

XRA A

INR A

STA 49AA

CALL 3980

LXI D, DC 54

CALL 4611

CALL 3980

MOV A, m

CP1 3101  
TANZ 3101

CALL 3980

1000

37A 1205R

CALL 2AF9

2B 5A 2B 2A

and

TRA A

OUT 1

4608

real

go to next char (hopefully a comma)

Syntax error w/ arrow.

max error w/ arrow.  
go to next real char (the string char)

*[Handwritten signature]*



# PUT + GET

Syntax = PUT filename, tapemode, one variable or constant (CR/L)

GET filename, tapemode, one variable (CR/L)

21 XX XX PUT (49B0) LXI H, PUTMSG Press RETURN when ready to record DATA "

↓

CD XX XX

21 XX XX

22 22 25 00

CD F9 2A

21 7F 00

22 22 25 00

X

13 06 30

CALL BOTH RTN

LXI H, PUTWH1

SHLD WH1 + 1

CALL 2AF9H

LXI H, 07FH

SHLD WH1 + 1

— record what's left —

JMP 3DD6H

21 XX XX GET (49CB) LXI H, GETMSG Press RETURN when ready to read DATA "

↓

CD XX XX

21 XX XX

22 21 0C

CD 0B 2C

21 BA 20

22 21 0C

X

13 06 30

CALL BOTH RTN

— read 1st record —

LXI H, GETWH0

SHLD WH0 + 1

CALL 'INPUT' rtn

LXI H, 20BAH

SHLD WH0 + 1

JMP 3DD6H

Changes →

mvi A, C3H  
STA 20BAH  
LXI H, GETWH0  
SHLD 20BCH

LXI H, EBF3H  
SHLD 20BAH  
mvi A, 2AH  
STA 20BCH



PUSH H  
LXI H, RETURN  
SHLD 0C18 *(KISRE)*

49E6

BOTH RTN

CALL 39B0H

incr to filename

LXI D, 0C54H

CALL 4611H

set up filename and PAB

CALL 39B0

incr to , after PAB hopefully

49FF=E1

MOV A, M

CPI ,

JNZ 21D1H

if no comma, "Syntax error"

incr to data

LXI H, BUFFER

SHLD BUFPTR

JMP 39B0H *ad use it's RET*

LXI H, COMMMSG  
CALL 39C9H

Press RETURN when ready to p'

49F9  
21 73 4A  
22 71 4A

E3 B0 39

CHGD

NOTCR  
POP CALL  
CALL WH0  
CPI CR  
JNZ NOTCR  
CALL WH1 *+ use RET*

PUTWH1

PUSH H

LHLD BUFPTR

MOV M, A

INR L

CZ Record it + ~~WH0~~ *LXI H, BUFFER / RET*

SHLD BUFPTR

POP H

RET

~~2500~~  
2637  
3CE9

BUFPTR EQU 4A7EH

BUFFER EQU 4A80H

RETURN

IN 0F2H  
CPI ESC  
JNZ 2070H  
LXI H, 07FH  
SHLD WH1H  
LXI H, 05F3H  
SHLD 20BAH  
MVI A, 02AH  
STA 20BCH  
LXI H, 2069H  
SHLD 0C18H  
JMP 20ADH *(ESC) RETN*

WILL HAVE PROBS  
IF I PRESS ESC  
WHILE a GET or  
PUT because it  
won't restore  
contents of  
20BA and  
WH1 as WH0



EC will replace LINE

EC means extended command

these commands will include:

HELP  
TRACE ON  
TRACE OFF  
APPEND file  
VARIABLES (for use with VarNames)  
etc

~~TRACE~~

~~REP~~

~~ca INX H / INX H / INX H~~

~~at 2542 and — with~~

~~CALL TRACE~~



both  $\langle \text{LIST} \rangle$  using variables and/or numbers  
RUN

To allow both LIST and RUN to use variables,  
change 3C9F and 3CA0 to Jxxxx, where xxxx is the  
address of the routine PARAMIT.

3C9E = CD 60 3C

|            |            |
|------------|------------|
| PARAMIT DS | PUSH D     |
| FS         | PUSH PSW   |
| CD B8 3B   | CALL PARAM |
| EB         | XCHG       |
| F1         | POP PSW    |
| D1         | POP D      |
| C9         | RET        |

What good this B for, I don't know.

3C9F is a Jxxxx, where xxxx is the  
address of the routine PARAMIT above.



# TRACE

replace 2542-2544 with CALL TRACE

TRACE    PUSH    P  
           PUSH    B  
           PUSH    D  
           PUSH    H

LXI    D, 48C2

CALL    24AF

LXI    H, 48C2

CALL    39C4

CALL    30DB

CALL    30D6 CROUT

POP    H

POP    D

POP    B

POP    P

INX    H  
 INX    H  
 INX    H

CALL    WHO here for line at a time executing

RET    to execute the line

print out  
line to be  
executed

caution - when BASIC RUNs,  
 it first checks every  
 line, so the whole program  
 will be Picked by TRACE,  
 and then it will execute

after a GOTO, trace prints line  
 right after the line which had a  
 GOTO, but executes the correct line



39BE 19  
 0D  
 CZ BE 39  
 C9

DAD D  
 DCR C  
 JNZ 39BE  
 RET

39C4 0E 0D  
 C3 CB 39

39C9 02 22

39CB 7E

47

B9

C8

FE 0D

01 CA DI 21

CD B9 3D

23

C3 CB 39

DB

MVI C, 13 (CR)

JMP 39CB

MVI C, "

MOV A, M

MOV B, A

CMP C

RZ

CPI CR

JZ Syntax error

CALL 30B9

INX H

JMP 39CB

1 10 FOR X = 120 TO 127

2,3 20 FOR Y = 40 TO 40 (X) 27F2 = CD

4,7,8 30 !X\*Y,

5,9,12,15 40 NEXT Y

6,10,13,16 50 NEXT X

11,14 = EXEC



## APPEND rtn

|      |       |               |
|------|-------|---------------|
| LHLD | 49ACH | start of text |
| SHLD | TEMP  |               |
| LHLD | 48AAH | end of text   |
| SHLD | 49ACH | start=end     |

then, do a normal LOAD

then, REN, with the first line being  
+10 of the last line of the 1st  
program

|      |       |
|------|-------|
| LHLD | TEMP  |
| SHLD | 49ACH |



Metamorphic Products announces

A Multi-Character-Variable-Name

Extension for Polymorphic Systems'

A00 BASIC, featuring

length of

\* 1 Variable Names of any length, limited only by input line length (normally 80 chars)

\* Upper and lower case Variable Names

~~\* NO INCREASE SPEED~~

\* DOES NOT SLOW EXECUTION AT ALL

\* The new command VARIABLES lists out all the current names in the Variable Table

\* 2 New Error <sup>messages</sup> added



## User's Guide

Do not use any of BASIC APPS reserved words as a variable name or embedded in reserved words or inside of other names or other characters -

i.e.

~~PRINT~~

REPRINTS will not work, <sup>because</sup> for <sup>25</sup> it contains the BASIC reserved word PRINT

Only up to 127 <sup>FF can't be used because it is EOT or maybe 128 if chg is made</sup> variable names

Only 1K of characters

maybe → (once a name is typed in + CR, a name can not be deleted w/o SCR.)

the best way is to use a <sup>uppercase</sup> capital letter for the first character of the variable, and then lower case for the rest of the letters. Do not hyphenate as the hyphen character means subtract in APPS Basic.

→ EX → Wages

→ EX → Overtime Hours <sup>for clarity in a multi word variable, capitalize the first character of each word.</sup>



## Notes on VarNames

Need to setup tape save, load, + verify routines to include VARIABLE, TOPVARPTR, and VarTable; i.e., put LXI H, VarTable at 4780H for SAVE

Add to start up message - 3DAC is routine to run it, and current message is at 2004

Need to Allow 1 char names <sup>UNIVERSAL MOVE ROUTINE</sup> (routines not Command SW)

Need to make SCR reset Variable, TOPVARPTR, and put FF's at VarTable and VarTable+1  
or 00 if chg is made

Need to disregard anything between '%' and ';' i.e. format statement !%C\$12F3, <sup>here</sup>

Need to disregard E for exponent in numbers i.e. 1.2345678E90

chg = \* Make 00 be end of Table, instead of FF



A1

Notes on Var Names

~~~~~

A1

PRINT

4B00 = FF

?
START OF
TEXT?

1	2	3	4	5	6	7	8	9	A	B	C	D	E
0E	0A	00	20	58	F5	CD	E0	33	30	37	36	29	00
Line	10	←	0	X	token	token	token						
length	1n	1n	0		= CALL	(	3	0	7	6	)	CR	

~~Pre Compression~~

~~Compressed~~

4B0F = 0B

F	10	11	12	13	14	15	16	17	18	19	1A
0B	14	00	20	92	9A	E0	35	30	29	00	01
line		←	0	token	token	token					
length	20	1n	0	CHR	(	5	0	)	CR	pgr and indicator	

W
BASIC 2 COMPRESSION ALGORITHM

0E	X	X	X	W	a	g	e	s	=	token	2	6	1	C _R
Len	2	3	20	4	1	3	F5	32	36	31	0D			
0B	X	X	X	A	1	=	token	2	6	1	C _R			
Len	2	3	20	4	1	3	F5	32	36	31	0D			
	1	2	3	4	5	6	7	8	9	A	B			

← precompression

VarNames
compression
results

← after compression

PROGRAM	'MULTI CHARACTER VARIABLE NAMES' PATCH	PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	SETUP	DATE	PUNCH						

STATEMENT																
Name		Operation		Operand		Comments										
1	8	10	14	16	20	25	30	35	40	45	50	55	60	65	70	75
SETUPEXP		MVI		A, 0C3H												
		STA		2405H												
		LXI		H, EXPAND												
		SHLD		2406H												
		LXI		H, 0												
		SHLD		2408H												
		SHLD		TEMP												
		LXI		H, VARIABLE+TABLENGTH												
		SHLD		205DH												
		LXI		H, VARIABLE												
		SHLD		TOPVARPTR												
		MVI		M, 0FFH												
		INX		H												
		MVI		M, 0FFH												
		MVI		A, 80H												
		STA		VARIABLE												
		LXI		H, SETUPEXP												
		SHLD		2001H												
SETUPVARI		LXI		H, 4056H												
		SHLD		2F39H												
		LXI		H, 4952H												

PUT A7H at 2F38H

← new start of text

DCX H
SALD TABLEND

For VARS

MVI A, 0A7H
STA 2F38H

* A standard card form, IBM electro 6509, is available for punching source statements from this form. Instructions for using this form are in any IBM System/360 assembler language reference manual. Address comments concerning this form to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden.

PROGRAM		MULTI CHARACTER VARIABLE NAMES		PATCH	
PROGRAMMER		SETUP		DATE	

STATEMENT

VarNames ⁹⁵¹ (P) ASM file via SMD

To Load - ~~SPJ 2000~~ to start up ASM

~~000000~~

PUSH RESET

Synthesizer TYPE → PVarNames to Load in VarNames

When done loading, ~~000~~ type CTL-Z

type L2FA0/J4BE1

L2FA6/J4BE1/I1U/SPJ2003/G

PROGRAM	'MULTI-CHARACTER VARIABLE NAMES' PATCH		PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	SETUP		DATE	PUNCH						

STATEMENT																		
1	Name	8	10	Operation	14	16	20	Operand	25	30	35	40	45	50	55	Comments	60	65
				SHLD			2F3BH											
				MVI			A, A7H											
				STA			2F3BH											
				LXI			H, VAR1											
				SHLD			2E49H											
	SETUP COMP			ORP			COMP											
				MVI			A, 0C3H											
				STA			24A1H											
				LXI			H, COMPRESS											
	JMP TO: 2041 WHEN DONE			SHLD			24A2											

* A standard card form, IBM electro 6509, is available for punching source statements from this form. Instructions for using this form are in any IBM System/360 assembler language reference manual. Address comments concerning this form to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden.

PROGRAM	MULTI-CHARACTER VARIABLE NAMES' PATCH	PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	New Command 'VARI'	DATE	PUNCH						

STATEMENT																		
1	Name	8	10	Operation	14	16	20	Operand	25	30	35	40	45	50	55	Comments	60	65
	VARI			LXI			H, VARIABLE+1											
	AGAIN			MOV			A, M											
				ORA			A											
				JM			CRETURN											
				CALL			WH1											
	MORE			INX			H											
				JMP			AGAIN											
	CRETURN			INR			A											
				PUSH			PSW											
				CALL			CROUT											
				POP			PSW											
				JNZ			MORE											
				RET														

PROGRAM	'MULTI CHARACTER VARIABLES' PATCH		PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	EXPAND		DATE	PUNCH						

STATEMENT																		
1	Name	8	10	Operation	14	16	20	Operand	25	30	35	40	45	50	55	Comments	60	65
EXPAND				CPI		CR												
				JZ		ZERO FLAG												
				CPI		"												
				JZ		CHG FLAG												
				STA		TEMP												
				LDA		FLAG												
				ORA		A												
				LDA		TEMP												
				JNZ		24DAH												
				CPI		8FH												
				JZ		SET FLAG												
				CPI		'@'												
				JC		24DAH												
				CPI		'E'												
				JNC		24DAH												
				SUI		41H												
				PUSH		D												
				MVI		D, 0												
				MOV		E, A												
				CALL		MULT BY 10												
				INX		H												
				MO														

* A standard card form, IBM electro 6509, is available for punching source statements from this form. Instructions for using this form are in any IBM System/360 assembler language reference manual. Address comments concerning this form to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden.

PROGRAM	'MULTI CHARACTER VARIABLES' PATCH		PUNCHING	GRAPHIC						
PROGRAMMER	EXPAND - continued		INSTRUCTIONS	PUNCH						
DATE										

STATEMENT																		
1	Name	8	10	Operation	14	16	20	Operand	25	30	35	40	45	50	55	Comments	60	65
				MOV		A, m												
				CPI		'/'												
				JNC		OK												
				CPI		':'												
				JC		OK												
	NOT OK			DCX		H												
				MOV		A, m												
				POP		D												
				JMP		24DAH												
	OK			ADI		50H												
				ADD		E												
				POP		D												
				PUSH		H												
				LXI		H, VARIABLE												
	FINDTOKE			STA		TEMP												
				MOV		A, m												
				CPI		0FFH												
				JZ		BADERROR												
				LOA		TEMP												
				CMP		M												
				INX		H												
				JNZ		FINDTOKE												
				JMP		24FDH												
	ENDTABLE			LXI		E, 2383H												

PROGRAM	MULTI-CHARACTER VARIABLE NAMES' PATCH	PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	EXPAND - continued		DATE	PUNCH					

[illegible]

* A standard card form, IBM electro 6509, is available for punching source statements from this form. Instructions for using this form are in any IBM System/360 assembler language reference manual. Address comments concerning this form to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden.

PROGRAM	'MULTI-CHARACTER VARIABLE NAMES' PATCH	PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	EXPAND UTILITIES TUNED STORAGE	DATE	PUNCH						

STATEMENT										Comments									
1	Name	8	10	Operation	14	16	20	Operand	25	30	35	40	45	50	55	60	65		
	ZEROFLAG			STA		TEMP													4
				JMP		ZERO													
	SETFLAG			STA		TEMP													
				MVI		A, 1													
				JMP		SET													
	CHGFLAG			STA		TEMP													
				LDA		FLAG													
				ORA		A													
				JNZ		ZERO													
				INR		A													
				JMP		SET													
	ZERO			XRA		A													
	SET			STA		FLAG													
	RETURN			LDA		TEMP													
				JMP		24DAH													
	MULTBY10			PUSH		H													
				MOV		H, D													
				MOV		L, E													
				DAD		H													
				DAD		H													
				DAD		D													



PROGRAM	MULTI-CHARACTER VARIABLE NAMES PATCH			PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	EXPAND UTILITIES - continued				DATE	PUNCH					

[illegible]

* A standard card form, IBM electro 6509, is available for punching source statements from this form. Instructions for using this form are in any IBM System/360 assembler language reference manual. Address comments concerning this form to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden.

LN 1440 → COMPRESS

PUSH H
MVI C, 3
STA FLAG
= E5

FLAG = 1 for quotes
FLAG = FF for REM

CMDRE

LXI H, 48C1H
INR C
INX H
MOV A, M
CPI CR
JZ CZEROFLAG
CPI 8FH
JNZ CNEXT
MVI A, 0FFH
JMP CSETIT
STA TEMP
LDA FLAG
ORA A
LDA TEMP
JM CMDRE

REM token

we're going to make FLAG minus
and non-zero

CNEXT

set flags

if FLAG set for REM - so we won't
bother anything inside of Quotes in
a REM line

PUSH PSW
CPI ''
JZ CCHGFLAG
POP PSW
JNZ CMDRE

save flags

if ''

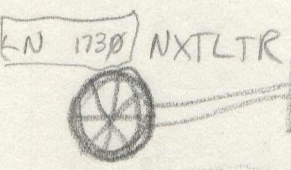
restore flags

CALL INPUTCHK
JC CMDRE

Carry set if A ≠ a legal LABEL char.
Save line char counter (in C)
Save current line position

legal Var char if it gets here →

PUSH B
PUSH H
MVI B, 0
MOV A, M
CALL INPUTCHK
JC CEND
INX H
INR B
JMP NXTLTR



ITSDK

CEND

DCR B
JM CHECKIT
CPI '0'
JC CHECKIT
CPI '9'
JNC CHECKIT
INR B
JMP ITSDK
CHECKIT INR B

can't have 1st
char be numeric

if < '0'

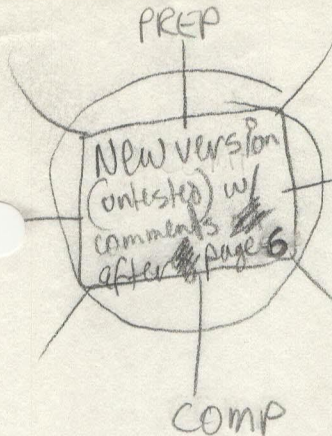
if > '9'

check for
second character numeric

put length of
LABEL into B and C
and STORE LABELEND

HL = end of LABEL + 1
put length into B and C

Restore HL back to start of LABEL



PUSH H
PUSH D
DCX D
LDAX D
CPI 0FFH
JZ Tablend
INX D
LDAX D
CMP m
JNZ No Match Yet
DCR C
INX H
INX D
JNZ COMP

Save HL for later too

DE = VarTable
HL = LABEL

see if at end of VarTable

1760



MAYBE

INX D
LDAX D
ORA A
JP No Match Yet

To make sure 'COUNT' will not match 'COUNTER' in VarTable.

Match

POP D
DCX D
LDAX D
JMP CONTINUE

Gets DE to TOKEN+1
Gets DE to the token
Gets token

Compare LABEL with VarTable and add LABEL if needed

No Match Yet

POP D
GOPAST INX D
LDAX D
ORA A
JP GOPAST
INX D - point to TOKEN+1

Gets DE at 1st char of a LABEL in VarTable.

To make sure 'COUNT' will not match 'RECOUNT' in VarTable.

POP H
JMP PREP

POP H
JMP PREP

Tablend

POP D
POP H
PUSH H

Restore HL to start of LABEL but save it for later too

FOR TOKEN BY PUTTING BY

LOA VARIABLE
ORA OVERFLOW
JZ OverflowError
INR A
STA VARIABLE
DCR A

- if variable = 0, then print error msg & restore stack.

gets new byte Token + puts (it+1) back

DE = LABEL
HL = VarTable

XCHG
LHLD TOPVARPTR
MOV M, A
PUSH PSW
INX H

put 'first char of LABEL' ptr into DE
get top of VarTable ptr - where ff is at
Store byte token
Save token byte

LDAX D
MOV M, A
INX D
DCR B
JNZ MOVMORE
INX HL
MVI M, 0FFH

PUT LABEL
INTO TABLE

B has char count of LABEL

move Label
to VarTable
and put ff at
end, and store
HL at TOPVARPTR

SHLD TOPVARPTR
POP PSW

Get token byte back

CONTINUE POP H
MVI GO
CAGAIN SUI 0AH
JM GOTIT
INR C
JMP CAGAIN
GOTIT ADI 3AH
MOV B, A
MOV A, C
ADI 41H

SUI 80H
= 10 decimal

A now has NBR in ASCII
B now has NBR
A has LTR
A now has LTR in ASCII

convert byte
token to
LTR and NBR

Doesn't
work

LTR is 12, 100 high
NBR is 8, 100 high

MOV M, A
MOV A, B
INX H
POP B

put LTR into M
put NBR into A

restore line char counter

INR C
MOV M, A
INX H
INR C

put NBR into M

PUSH H
PUSH B

XCHG
LHLD LABELEND
MOV A, M

DE = TOKEN END + 1
HL = LABEL END + 1

move the
chars from
HL on up to
DE; stop if
CR

MOVAM

STAX D
CPI CR
JZ BPOPH
INX H
INX D
JMP MOVAM

BPOPH

POP B

POP H

POP D

JMP CMORE + 2

INPUT CHK ORA A

JM SNOTOK

~~CPI INPUT CHARACTER~~~~OK INPUT~~

CPI '@'

RC

CPI 'E'

JM OKINPUT

CPI 'a'

RC

CPI 'Σ'

JNC SNOTOK

OKINPUT STC

CMC

RET

SNOTOK

STC

RET

CCHGFLAG

POP PSW

LDA FLAG

ORA A

JNZ CZERO

MVI A, 1

JMP CSETIT

CZERO XRA A

CSETIT STA FLAG

JMP CMORE

CZERO FLAG

CREND

XRA A

STA FLAG

XCHG

POP H

MOV M, C

POP PSW

RET

INX

H

MOA

MVA

X

MVA

- 2 for

p/h for

- 2 for

b/c to job of non-100%

3540 to 4086

4085 = 01 and

4

X 1 0
 EX 1 0
 MOV M, A
 INX H

- get ptr to top of VarTable (value ff is)
 - store byte token

HC82=07 x 09

32H 10 HC82

2WB CWOKE
 CSEED 214 674
 CSEED X84 A
 2WB CSEED
 1W1 A7
 215 CSEED
 084 A
 104 674
 606 62M
 CCH8E8E

15E1
 21C
 15E1
 CWC
 0X1061 21C
 21C 210X
 CBI ,E,
 15C
 CBI ,0,
 2W 0X1061
 CBI ,E,
 15C
 CBI ,0,
~~21C 0X1061~~
~~CBI ,0,
 2W 0X1061~~
 2W 210X
 104 A
 2WB CWOKE + 5
 606 H
 606 B

15E1
 606 62M
 104 10C
 606 H
 X84 e
 214 674
 X84 A
 CSEED
 CSEED 674

104 A
 2WB CWOKE + 5
 606 H
 606 B



PROGRAM				PUNCHING INSTRUCTIONS	GRAPHIC							
PROGRAMMER		Compress Utilities	DATE		PUNCH							

Name	Operation	Operand	Comments
OVERFLWD	LXI	H, ERRORMSG	'Variable "Table"'
	CALL	39C9H	
	LXI	H, ERRORMSG+9	'Table'
	CALL	39C9H	
	LXI	B, 237FH	'full'
	JMP	222DH or further on	'error'

* A standard card form, IBM electro 6509, is available for punching source statements from this form. Instructions for using this form are in any IBM System/360 assembler language reference manual. Address comments concerning this form to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden.

THIS

PROGRAM	MULTI CHARACTER VARIABLES' PATCH	PUNCHING INSTRUCTIONS	GRAPHIC						
PROGRAMMER	TEMPORARY STORAGE	DATE	PUNCH						

NAME										OPERATION										OPERAND										STATEMENT										COMMENTS																								
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65
TEMP										DB										0																																												
FLAG										DB										0																																												
TOPVARPTR										DW										VARIABLE																																												
VARIABLE										DB										80H																																												
VARIABLE										DB										7 6 5 4 3 2 1 0																																												
ERRORMSG										ASC										'Variable Table'																																												
VARIABLE										DB										80H																																												
TOPVARPTR										DW										VARIABLE																																												
LABELEND										DW																																																						
TABLEND										DS										2																																												
VARIABLE										DS										TABLENGTH																																												

* A standard card form, IBM electro 6509, is available for punching source statements from this form. Instructions for using this form are in any IBM System/360 assembler language reference manual. Address comments concerning this form to IBM Nordic Laboratory, Publications Development, Box 962, S-181 09 Lidingö 9, Sweden.

9

FIND LABEL, generate new one if needed
 REPLACE LABEL with a variable TOKEN $\{A0, A1, A2, \dots, Z7, Z8, Z9\}$

at START, HL = start of LABEL in input line after BASIC Token compression
 DE = start of Variable + 1
 BndC = length of LABEL

PREP	PUSH	H	Save start of LABEL
	PUSH	D	Save current position in Variable
	DCX	D	Dec to token
	LDAX	D	Get the byte
	CPI	0FFH	see if end of table
	JZ	TABLEND	if no match found
	INX	D	incr back to first char of a Variable label
COMP	LDAX	D	get a Variable char
	CMP	M	is (HL) = (DE)?
	JNZ	NOMCHYET	if not, then this is not the label
	DCR	C	decrement LABEL char counter
	INX	H	incr LABEL pointer
	INX	D	incr Variable pointer
	JNZ	COMP	if not yet at end of LABEL
MAYBE	LDAX	D	check next char
	ORA	A	set flags
	JP	NOMCHYET	if not a token
MATCH	POP	D	Gets DE to token + 1
	DCX	D	DE now points to token
	LDAX	D	get token
	JMP	CONTINUE	and figure the two char version for this LABEL
NOMCHYET	POP	D	Gets DE to token + 1 of current Variable LABEL
GOPAST	INX	D	incr to next Variable char
	LDAX	D	get the char
	ORA	A	set flags
	JP	GOPAST	if not a token
	INX	D	go past the token
	POP	H	get the start of LABEL back into HL
	JMP	PREP	and check next Variable LABEL
TABLEND	POP	D	restore the stack
	POP	H	and especially HL
	PUSH	H	but save HL for later
PVOTOKENB	LDA	VARIABLE	get variable counter
	ORA	A	set flags
	JZ	OVERFLWD	if it = 0, this makes too many labels

see if at end of Variable
 to make sure 'COUNT' will NOT match w/ 'COUNTER'.
 to make sure COUNT will NOT match COUNTER.
 make ready to generate a new token and Variable entry

	INR	A	otherwise, increment for the next VarName
	STA	VARIABLE	and store it
	DCR	A	but reset it for current token
	XCHG		now DE points to first char of LABEL
	LHLD	TOPVARPTR	get address of 0FFH (end of Vartable)
	MOV	m, A	store the new token
	PUSH	PSW	and save it for expansion later
MOVEMORE	INX	H	increment to next byte
	CALL	CMPREND	see if at end of table's memory
	JZ	OVERFLWD	if we overflowed
	LDAX	D	otherwise get next char of LABEL
	MOV	m, A	and move it to Vartable's next position
	INX	D	incr LABEL pointer
	DCR	B	dec LABEL char pointer
	JNZ	MOVEMORE	if not, yet all moved
	INX	H	incr to next byte in table after end of LABEL
	CALL	CMR	check if overflow just in case
	JZ	OVERFLWD	if we did go over
	MVI	m, 0FFH	otherwise put in end of table char
	SHLD	TOPVARPTR	and store the addr of it
	POP	PSW	restore the token
CONTINUE	POP	H	get addr of LABEL in line
	SUI	80H	subtract 80H from TOKEN
	MVI	C, 0	zero C
CAGAIN	SUI	0AH	subtract 10 from token
	JM	GOTIT	if we underflowed
	INR	C	increment the counter
	JMP	CAGAIN	and do it again
GOTIT	ADI	03AH	restore 10 and make A an ASCII number
	MOV	B, A	put the number into B
	MOV	A, C	A now has the 'almost' Letter
	ADI	41H	now A is an ASCII letter
	MOV	m, A	put letter at first char of LABEL
	MOV	A, B	now A = the number
	INX	H	increment to next char of LABEL
	POP	B	restore LABEL char counter in C (B=0)
	PUSH	PSW	save NBR in A
	MOV	A, C	put char count into A
	CPI	1	was it one?

$$C = \frac{\text{TOKEN} - 80}{10}$$


```
JZ    PUSHOUT1
CPI    2
JNZ    PULLIN
GROP   POP    PSW
      MOV    M,A
      JMP    BPOPH
PULLIN1 MOV    A,M
AGIN   INX    H
      MOV    B,M
      MOV    M,A
      CPI    CR
      JZ     PEND
      MOV    A,B
      JMP    AGIN
PEND   LHLD   LABELEND
      JMP    GROP
```

if it was a 1 char label
A=Z?
if it was more than a 2 char label
restore NBR
and put it into line
jmp to end if up
put char into B

} move line over 1 byte

	MAR A	otherwise, increment for the next var name
	STA VARIABLE	and store it
	DCR A	and reset it to the current token
	XCHG	now DE points to first char of LABEL
	LHLD TOPVARPTR	get address of ff (end of table)
	MOV M, A	store the token
	PUSH PSW	and save it for expansion later
MOV MORE	INX H	increment to next byte
	LDAX D	get a char of LABEL
	MOV M, A	and move it to Variable
	INX D	incr LABEL pointer
	DCR B	dec LABEL char pointer
	JNZ MOV MORE	if not yet done
	INX H	incr to next char after end of LABEL in Variable
	MVI M, 0FFH	and put end of table char in

MOVE LABEL TO TABLE (might be able to replace it with Universal move RTN)

CMPREND

PUSH D
XCHG
LHLD TABLEND
CALL CMPR
XCHG
POP D
RET

CMPRS HL
WITH CONTENTS
OF TABLEND
Z set if equal

PULL IN 1

move line down one char to make room for NBR